Informe de Laboratorio 03

Tema: Arreglos de Objetos

Nota

Estudiante	Escuela	Asignatura
Hernan Andy Choquehuanca Zapana hchoquehuancaz@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la Programación II Semestre: II Código: 20232191

Laboratorio	Tema	Duración
03	Arreglos de Objetos	02 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 20 Septiembre 2023	Al 25 Septiembre 2023

1. Tarea

- Analice, complete y pruebe el Código de la clase DemoBatalla
- Solucionar la Actividad 4 de la Práctica 1 pero usando arreglo de objetos
- Solucionar la Actividad 5 de la Práctica 1 pero usando arreglos de objetos
- Utilizar Git para evidenciar su trabajo.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Pro 22H2 64 bits.
- VIM 9.0.
- Visual Studio Code
- Git 2.41.1.
- Cuenta en GitHub con el correo institucional.
- Variables Simples
- Arreglos de Objetos
- Métodos

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/hernanchoquehuanca/fp2-23b.git`
- URL para el laboratorio 03 en el Repositorio GitHub.
- `https://github.com/hernanchoquehuanca/fp2-23b/tree/main/fase01/lab03`

4. Actividades con el repositorio GitHub

4.1. Commits

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

- Primero acomodamos y copiamos el código para luego empezar a completarlo
- El código fue el siguiente:

Listing 1: DemoBatalla.java

```java
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("\nNaves creadas:");
mostrarNaves(misNaves);
mostrarPorNombre(misNaves);
mostrarPorPuntos(misNaves);
System.out.println("\nNave con mayor nmero de puntos: " + mostrarMayorPuntos(misNaves));
}
// Mtodo  para mostrar todas las naves
```

```
35 public static void mostrarNaves(Nave [] flota){
36 }
37 // Mtodo  para mostrar todas las naves de un nombre que se pide por teclado
38 public static void mostrarPorNombre(Nave [] flota){
39 }
40 // Mtodo  para mostrar todas las naves con un nmero de puntos inferior o igual
41 //al  nmero  de puntos que se pide por teclado
42 public static void mostrarPorPuntos(Nave [] flota){
43 }
44 // Mtodo  que devuelve la Nave con mayor nmero de Puntos
45 public static Nave mostrarMayorPuntos(Nave [] flota){
46 }
47 //Crear un mtodo que devuelva un nuevo arreglo de objetos con todos los objetos previamente
       ingresados //pero aleatoriamente desordenados
48 }
```

Listing 2: Nave.java

```
1  public class Nave {
2  private String nombre;
3  private int fila;
4  private String columna;
5  private boolean estado;
6  private int puntos;
7  // Metodos mutadores
8  public void setNombre( String n){
9  nombre = n;
10 }
11 public void setFila(int f){
12 fila = f;
13 }
14 public void setColumna(String c){
15 columna = c;
16 }
17 public void setEstado(boolean e){
18 estado = e;
19 }
20 public void setPuntos(int p){
21 puntos = p;
22 }
23 // Metodos accesores
24 public String getNombre(){
25 return nombre;
26 }
27 public int getFila(){
28 return fila;
29 }
30 public String getColumna(){
31 return columna;
32 }
33 public boolean getEstado(){
34 return estado;
35 }
36 public int getPuntos(){
37 return puntos;
38 }
```

```
39 // Completar con otros mtodos necesarios
40 }
```

Listing 3: Commit: Agregando las dos clases (DemoBatalla y Nave) para completar los métodos

```
$ git add .
$ git commit -m "Agregando las dos clases (DemoBatalla y Nave) para completar los
    mtodos"
$ git push -u origin main
```

- Realizamos el método mostrarPorPuntos usando que mostrará aquellas naves menores o iguales a los puntos ingresados.
- Además creamos un método llamado mostrarNave, la cual nos servirá para los futuros métodos y este.
- El método fue el siguiente:

Listing 4: DemoBatalla.java

```
1  // Mtodo  para mostrar todas las naves con un nmero de puntos inferior o igual
2  //al  nmero  de puntos que se pide por teclado
3  public static void mostrarPorPuntos(Nave [] flota){
4    Scanner sc = new Scanner(System.in);
5    int puntos = sc.nextInt();
6
7    for (int i = 0; i < flota.length; i++){
8      if (flota[i].getPuntos() <= puntos){
9        mostrarNave(flota[i]);
10     }
11   }
12 }
```

Listing 5: Commit: Implementando el método mostrar por puntos, creando un nuevo método para mostrar naves (mostrarNave)

```
$ git add .
$ git commit -m "Implementando el mtodo mostrar por puntos, creando un nuevo mtodo para
    mostrar naves (mostrarNave)"
$ git push -u origin main
```

- Implementando el método mostrar por nombre, haciendo uso de un bucle for para luego comparar todos los elementos de flota, mostrando con el método mostrarNaver a aquellos que coincidan con el nombre ingresado.
- El método fue el siguiente:

Listing 6: DemoBatalla.java

```
1  // Mtodo  para mostrar todas las naves de un nombre que se pide por teclado
2  public static void mostrarPorNombre(Nave [] flota){
3    Scanner sc = new Scanner(System.in);
4    String name = sc.next();
```

```
5
6     for (int i = 0; i < flota.length; i++){
7       if (flota[i].getNombre().equals(name))
8         mostrarNave(flota[i]);
9     }
10  }
```

Listing 7: Commit: Implementando el metodo mostrarPorNombre haciendo uso de un bucle for y el método mostrarNave

```
$ git add .
$ git commit -m "Implementando el mtodo mostrarPorNombre haciendo uso de un bucle for y
    el metodo mostrarNave"
$ git push -u origin main
```

- Implementando el método mostrarMayorPuntos haciendo uso de variables auxiliares que permiten guardar datos enteros como el número máximo y su posición en el arreglo de naves.
- El método fue el siguiente:

Listing 8: DemoBatalla.java

```
1   // Mtodo  que devuelve la Nave con mayor nmero de Puntos
2   public static Nave mostrarMayorPuntos(Nave [] flota){
3     int min = flota[0].getPuntos();
4     int positionNave = 0;
5     for (int i = 1; i < flota.length; i++){
6       if (flota[i].getPuntos() > min){
7         min = flota[i].getPuntos();
8         positionNave = i;
9       }
10    }
11    return flota[positionNave];
12  }
```

Listing 9: Commit: Implementando el método mostrarMayorPuntos haciendo uso de variables auxiliares para guardar valores de puntaje y posición

```
$ git add .
$ git commit -m "Implementando el mtodo mostrarMayorPuntos haciendo uso de variables
    auxiliares para guardar valores de puntaje y posicin"
$ git push -u origin main
```

- Implementando el método mostrarNaves simplemente usando un bucle for para recorrer por el arreglo y dentro utilizando el futuro método mostrarNave.
- El método fue el siguiente:

Listing 10: DemoBatalla.java

```
1   // Mtodo  para mostrar todas las naves
2   public static void mostrarNaves(Nave [] flota){
3     for (int i = 0; i < flota.length; i++){
```

```
    mostrarNave(flota[i]);
  }
}
```

Listing 11: Commit: Implementando el método mostrarNaves, utilizando el futuro método mostrarNave

```
$ git add .
$ git commit -m "Implementando el mtodo mostrarNaves, utilizando el futuro metodo
    mostrarNave"
$ git push -u origin main
```

- Agregando comentarios a los métodos, para de alguna manera separarlos al imprimirlos dentro del main.
- Implementando el método desordenarFlota, utilizando Math.random para generar posiciones aleatorias.
- Para evitar las posiciones repetidas se agrego un bucle for para generar posiciones random hasta que se encuentre una vacía.
- El método fue el siguiente:

Listing 12: DemoBatalla.java

```
//Crear un mtodo que devuelva un nuevo arreglo de objetos con todos los objetos
    previamente ingresados
//pero aleatoriamente desordenados
public static Nave [] desordenarFlota(Nave flota []){
  Nave [] flotaRandom = new Nave [flota.length];
  for (int i = 0; i < flota.length; i++) {
    int r = (int)Math.random() * 10;
    while (flotaRandom[r] != null){
      r = (int)Math.random() * 10;
    }
    flotaRandom [r] = flota[i];
  }

  return flotaRandom;
}
```

Listing 13: Commit: Implementando el método para retornar un arreglo aleatoriamente desordenado

```
$ git add .
$ git commit -m "Implementando el mtodo para retornar un arreglo aleatoriamente
    desordenado"
$ git push -u origin main
```

- Finalmente implementamos el método mostrarNave, el cual está presente en los demás métodos.
- El método imprimirá el nombre, estado y puntos.
- El método fue el siguiente:

Listing 14: DemoBatalla.java

```java
public static void mostrarNave(Nave nave){
  System.out.println("\n Nave: " + nave.getNombre());
  if (nave.getEstado())
    System.out.println("Estado : Alive");
    else
    System.out.println("Estado : Dead");
  System.out.println("Puntos : " + nave.getPuntos());

}
```

Listing 15: Commit: Implementando el método mostrarNave, el cual está presente en otros métodos

```
$ git add .
$ git commit -m "Implementando el mtodo mostrarNave, el cual est presente en otros
    mtodos"
$ git push -u origin main
```

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

- Primero acomodamos y copiamos el código para luego empezar a editarlo.
- Reajustando variables y ahora haciendo uso de la nueva clase Soldier.
- El código fue el siguiente:

Listing 16: Ejercicio01.java

```java
import java.util.Scanner;

public class Ejercicio01 {
  //Solucionar la Actividad 4 de la Prctica 1 pero usando arreglo de objetos
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);

    Soldier [] soldiers = new Soldier[5];

    for (int i = 0; i < soldiers.length; i++) {
      System.out.print("Ingrese el nombre del soldado Nro " + (i + 1) + " : ");
      soldiers[i].setNombre(sc.next());

      System.out.print("Vida: ");
      soldiers[i].setVida(sc.nextInt());
    }

    for (int i = 0; i < soldiers.length; i++) {
      System.out.println("Soldado Nro " + (i + 1) + ": " + soldiers[i].getNombre() + " -
          vida: " + soldiers[i].getVida());
    }
  }
}
```

Listing 17: Soldier01.java

```java
class Soldier {
  private int vida;

  public void setVida(int v){
    vida = v;
  }
  public int getVida(){
    return vida;
  }
}
```

Listing 18: Commit: Ejercicio01 terminado, creando otra clase llamada Soldier para los objetos de soldados

```
    $ git add .
    $ git commit -m "Ejercicio01 terminado, creando otra clase llamada Soldier para los
        objetos de soldados"
    $ git push -u origin main
```

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

- Primero acomodamos y copiamos el código para luego empezar a completarlo
- Reajustando variables y ahora haciendo uso de la nueva clase Soldier.
- El código fue el siguiente:

Listing 19: Ejercicio02.java

```java
public class Ejercicio02 {
  /*
    escribir un programa donde se creen 2 ejrcitos, cada uno con un nmero aleatorio de
        soldados entre
    1 y 5, considerando slo su nombre. Sus datos se inicializan automticamente con nombres
        tales como Soldado0,
      Soldado1   , etc. Luego de crear los 2 ejrcitos se deben mostrar los datos de todos los
        soldados de ambos ejrcitos
    e indicar qu  ejrcito  fue el ganador.
    Restriccin : aplicar arreglos estndar y mtodos para inicializar los ejrcitos, mostrar
        ejrcito y mostrar ejrcito
    ganador. La mtrica a aplicar para indicar el ganador es el mayor nmero de soldados de
        cada ejrcito, puede
    haber empates. (Todava no aplicar arreglo de objetos)
   */

   public static void main(String[] args) {
    Soldier[] army1 = createArmy();
    Soldier[] army2 = createArmy();

    System.out.println("Ejercito Nro 1");
    showArmy(army1);
    System.out.println("\nEjercito Nro 2" );
```

```
    showArmy(army2);

    armyWinner(army1, army2);
  }

  public static Soldier[] createArmy() {
    int n = (int) (Math.random() * 5) + 1;

    Soldier [] army = new Soldier[n];
    for (int i = 0; i < n; i++) {
      army[i].setNombre("Soldado" + i);
    }

    return army;
  }

  public static void showArmy(Soldier[] army) {
    for (int i = 0; i < army.length; i++) {
      System.out.println(army[i].getNombre());
    }
  }

  public static void armyWinner(Soldier[] army1, Soldier[] army2) {
    if (army1.length > army2.length) {
      System.out.println("El ganador es el Ejercito Nro 1");
    } else if (army2.length > army1.length) {
      System.out.println("El ganador es el Ejercito Nro 2");
    } else {
      System.out.println("Es un empate entre los ejercitos");
    }
  }
}
```

Listing 20: Soldier02.java

```
class Soldier {
  private int vida;
  private String nombre;

  public void setVida(int v){
    vida = v;
  }
  public int getVida(){
    return vida;
  }

  public void setNombre( String n){
    nombre = n;
  }
  public String getNombre(){
    return nombre;
  }
}
```

Listing 21: Commit: Reemplazando instanciaciones y demás se acomodó el código para que utilice arrays de Soldier en lugar de String

```
$ git add .
$ git commit -m "Reemplazando instanciaciones y dems se acomod el cdigo para que
    utilice arrays de Soldier en lugar de String"
$ git push -u origin main
```

4.2. Estructura de laboratorio 03

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

```
lab03
     DemoBatalla.java
     Ejercicio01.java
     Ejercicio02.java
     Nave.java
     Soldier.java

+------latex
         Informe_Lab03.pdf
         Informe_Lab03.tex

    |------img
             logo_abet.png
             logo_episunsa.png
             logo_unsa.jpg
             prueba01.png

    +------src
           DemoBatallaInicial.java
           Ejercicio01.java
           Ejercicio02.java
           m1.java
           m2.java
           m3.java
           m4.java
           m5.java
           m6.java
           NaveInicial.java
           Soldier01.java
           Soldier02.java
```

5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

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

5.2. 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 aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: 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

Tabla 3: 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			
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	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		15	

6. Referencias

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html
- https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html
- https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html