Informe de Laboratorio 03

Tema: Arreglos de objetos

Nota

Estudiante	Escuela	Asignatura
Ryan Fabian Valdivia Segovia rvaldiviase@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 Septiembre 2023	Al 24 Septiembre 2023

1. Tarea

1.1. Actividad 1: Demo Batalla

- Analice, complete y pruebe el Código de la clase DemoBatalla.

1.2. Actividad 2

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

1.3. Actividad 3

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

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Home Single Language 64 bits 22621.2283
- VIM 9.0.
- Visual Studio Code 64 bits 1.82.2
- OpenJDK 64-Bits 11.0.16.1

- Git 2.41.0.windows.1
- Cuenta en GitHub con el correo institucional.
- Código parcial proporcionado por el profesor.

3. URL de Repositorio Github

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

4. Actividades

4.1. Actividad 1

- Realicé un commit copiando el código parcial que nos proporcionó el profesor.

Listing 1: Comentando el código paricial

```
$ git log Ahorcado.java
commit b1ca8e2e60d9546da50d357333768c59ebeb0135
Author: RYAN VALDIVIA <rvaldiviase@unsa.edu.pe>
Date:  Sun Sep 24 14:36:27 2023 -0500
   Actividad 1: Copiando el codigo parcial que nos dio el profesor
```

Listing 2: Código parcial de DemoBatalla

```
6  import java.util.*;
7
8  public class DemoBatalla {
9      public static void main(String[] args) {
10         Nave[] misNaves = new Nave[10];
11         Scanner sc = new Scanner(System.in);
12         String nomb, col;
13         int fil, punt;
14         boolean est;
15         for (int i = 0; i < misNaves.length; i++) {
16             System.out.println("Nave " + (i + 1));
17             System.out.println("Nombre: ");
18             nomb = sc.next();
19             System.out.println("Fila ");
20             fil = sc.nextInt();
21             System.out.println("Columna: ");
22             col = sc.next();
23             System.out.println("Estado: ");
24             est = sc.nextBoolean();
25             System.out.println("Puntos: ");
26             punt = sc.nextInt();
27             misNaves[i] = new Nave(); // Se crea un objeto Nave y se asigna su referencia a
                   misNaves
```

```
28              misNaves[i].setNombre(nomb);
29              misNaves[i].setFila(fil);
30              misNaves[i].setColumna(col);
31              misNaves[i].setEstado(est);
32              misNaves[i].setPuntos(punt);
33          }
34          System.out.println("\nNaves creadas:");
35          mostrarNaves(misNaves);
36          mostrarPorNombre(misNaves);
37          mostrarPorPuntos(misNaves);
38          System.out.println("\nNave con mayor numero de puntos: " +
                mostrarMayorPuntos(misNaves));
39      }
40
41      // Metodo para mostrar todas las naves
42      public static void mostrarNaves(Nave[] flota) {
43      }
44
45      // Metodo para mostrar todas las naves de un nombre que se pide por teclado
46      public static void mostrarPorNombre(Nave[] flota) {
47      }
48
49      // Metodo para mostrar todas las naves con un numero de puntos inferior o igual
50      // al numero de puntos que se pide por teclado
51      public static void mostrarPorPuntos(Nave[] flota) {
52      }
53
54      // Metodo que devuelve la Nave con mayor numero de Puntos
55      public static Nave mostrarMayorPuntos(Nave[] flota) {
56      }
57      // Crear un metodo que devuelva un nuevo arreglo de objetos con todos los
58      // objetos previamente ingresados
59      // pero aleatoriamente desordenados
60  }
```

- También tenemos el código de la clase auxiliar Nave.

Listing 3: Clase Nave

```
61  public class Nave {
62      private String nombre;
63      private int fila;
64      private String columna;
65      private boolean estado;
66      private int puntos;
67
68      // Metodos mutadores
69      public void setNombre(String n) {
70          nombre = n;
71      }
72
73      public void setFila(int f) {
74          fila = f;
75      }
76
```

```
77      public void setColumna(String c) {
78          columna = c;
79      }
80
81      public void setEstado(boolean e) {
82          estado = e;
83      }
84
85      public void setPuntos(int p) {
86          puntos = p;
87      }
88
89      // Metodos accesores
90      public String getNombre() {
91          return nombre;
92
93      }
94
95      public int getFila() {
96          return fila;
97
98      }
99
100     public String getColumna() {
101         return columna;
102
103     }
104
105     public boolean getEstado() {
106         return estado;
107
108     }
109
110     public int getPuntos() {
111         return puntos;
112
113     }
114     // Completar con otros metodos necesarios
115 }
```

- Lo primero que hice fue elaborar el método de mostrarNaves, haciendo que simplemente se impriman todas las naves del arreglo junto con sus respectivos atributos, usando un ciclo for each para recorrer el arreglo.

Listing 4: Mostrar naves

```
116     public static void mostrarNaves(Nave[] flota) {
117         int i = 1;
118         for (Nave n : flota) {
119             System.out.println("Nave " + i + ":" + n.getNombre());
120             System.out.println("Posicion: " + n.getFila() + n.getColumna());
121             System.out.println("Puntos: " + n.getPuntos());
122             if (n.getEstado()) {
123                 System.out.println("Sigue con vida");
124             } else {
```

```
125             System.out.println("Fue destruido");
126         }
127         System.out.println();
128         i++;
129     }
130  }
```

- La razón por la trabajé este método primero, fue para usarlo de base para los siguientes métodos.
- Para el método de mostrarPorNombre, seguí la lógica del método anterior, con la modificación de mostrar solo las naves que coincidan con el nombre que ingrese el usuario.

Listing 5: Mostrar por Nombre

```
131  public static void mostrarPorNombre(Nave[] flota) {
132      Scanner sc = new Scanner(System.in);
133      String nombre = sc.next();
134      for (int i = 0; i < flota.length; i++) {
135          if (flota[i].getNombre().equals(nombre)) {
136              Nave n = flota[i];
137              System.out.println(n.getNombre());
138              System.out.println("Posicion: " + n.getFila() + n.getColumna());
139              System.out.println("Puntos: " + n.getPuntos());
140              if (n.getEstado()) {
141                  System.out.println("Sigue con vida");
142              } else {
143                  System.out.println("Fue destruido");
144              }
145          }
146      }
147  }
```

- Para el siguiente método, continué con la lógica, solo modificando el ciclo con un condicional para imprimir las naves que tengan un puntaje menor o igual al ingresado por teclado.

Listing 6: Mostrando por puntos menores o iguales

```
148  public static void mostrarPorPuntos(Nave[] flota) {
149      Scanner sc = new Scanner(System.in);
150      int puntos = sc.nextInt();
151      for (int i = 0; i < flota.length; i++) {
152          if (flota[i].getPuntos() <= puntos) {
153              Nave n = flota[i];
154              System.out.println("Nave " + i + ":" + n.getNombre());
155              System.out.println("Posicion: " + n.getFila() + n.getColumna());
156              System.out.println("Puntos: " + n.getPuntos());
157              if (n.getEstado()) {
158                  System.out.println("Sigue con vida");
159              } else {
160                  System.out.println("Fue destruido");
161              }
162          }
163      }
164  }
```

- El siguiente método es un algoritmo conocido y simple para saber cual es el objeto de mayor valor en un arreglo, lo usé para el método de mostrarMayorPuntos.

Listing 7: Mostrar la nave con mayor cantidad de puntos

```
165 public static Nave mostrarMayorPuntos(Nave[] flota) {
166     int mayor = 0;
167     for (int i = 0; i < flota.length; i++) {
168         if (flota[i].getPuntos() > flota[mayor].getPuntos()) {
169             mayor = i;
170         }
171     }
172     return flota[mayor];
173 }
```

- Finalmente, se viene algo un poco más complicado, el método para desordenar el arreglo de naves, para hacer esto, hice dos métodos auxiliares.
- En primer lugar, mi estrategia fue crear un arreglo de enteros conteniendo los numeros del 0 a la longitud del arreglo de naves (En este caso, 10), y usarlo para desordenar el arreglo.

Listing 8: Método para obtener números desordenados

```
174 public static int[] numerosRandom(Nave[] flota) {
175     int[] nums = new int[flota.length];
176     for (int i = 0; i < nums.length; i++) {
177         nums[i] = nums.length;
178     }
179     for (int i = 0; i < flota.length; i++) {
180         int n;
181         do {
182             n = (int) (Math.random() * flota.length);
183         } while (estaEnArreglo(nums, n, i));
184         nums[i] = n;
185     }
186     return nums;
187 }
```

- Y para esto, necesitaba una forma de comprobar que el número aleatorio que generé no estuviera antes en el arreglo, para que no haya números repetidos, por tanto, creé el método estaEnArreglo para hacer esa comprobación.

Listing 9: Método está en Arreglo

```
188 public static boolean estaEnArreglo(int[] arreglo, int num, int indice) {
189     for (int i = 0; i < indice; i++) {
190         if (arreglo[i] == num) {
191             return true;
192         }
193     }
194     return false;
195 }
```

- Con el arreglo de números aleatorios ya hecho, solo me queda armar todo para desordenar un arreglo de naves dado.

Listing 10: Desordenar arreglo

```
196 public static Nave[] desordenar(Nave[] flota) {
197     Nave[] desordenado = new Nave[flota.length];
198     int[] desorden = numerosRandom(flota);
199     for (int i = 0; i < desordenado.length; i++) {
200         desordenado[i] = flota[desorden[i]];
201     }
202     return desordenado;
203 }
```

- Solo me falta añadir un par de cosas en el método main para que funcione al 100

Listing 11: Método main

```
204 System.out.println("\nNave con mayor numero de puntos: " +
        mostrarMayorPuntos(misNaves).getNombre());
205     mostrarNaves(desordenar(misNaves));
```

- Con esto ya tengo el código de DemoBatalla listo para su uso.
- A continuación tengo algunas capturas de la ejecución del código, usando solo cinco naves en el arreglo para no tener que introducir tantos datos.
- Llenando los datos del arreglo.

```
PS C:\Users\usuario\Documents\workspace\fp2-23b\fase01\lab03> javac .\DemoBatalla.java .\Nave.java
PS C:\Users\usuario\Documents\workspace\fp2-23b\fase01\lab03> java DemoBatalla
Nave 1
Nombre:
Whisky
Fila
1
Columna:
C
Estado:
true
Puntos:
70
Nave 2
Nombre:
Indio
Fila
2
Columna:
C
Estado:
false
Puntos:
40
```

```
Nave 3
Nombre:
Fire
Fila
4
Columna:
J
Estado:
true
Puntos:
89
Nave 4
Nombre:
Mike
Fila
5
Columna:
D
Estado:
true
Puntos:
65
```

```
Nave 5
Nombre:
Delta
Fila
2
Columna:
A
Estado:
false
Puntos:
43
```

- Probando el método para mostrar todas las naves.

```
Naves creadas:
Nave 1:Whisky
Posicion: 1C
Puntos: 70
Sigue con vida

Nave 2:Indio
Posicion: 2C
Puntos: 40
Fue destruido

Nave 3:Fire
Posicion: 4J
Puntos: 89
Sigue con vida

Nave 4:Mike
Posicion: 5D
Puntos: 65
Sigue con vida

Nave 5:Delta
Posicion: 2A
Puntos: 43
Fue destruido
```

- Probando el método para búsqueda por nombre.

```
Delta
Delta
Posicion: 2A
Puntos: 43
Fue destruido
```

- Probando el método para búsqueda por puntos.

```
60
Nave 1:Indio
Posicion: 2C
Puntos: 40
Fue destruido
Nave 4:Delta
Posicion: 2A
Puntos: 43
Fue destruido
```

- Probando el método para saber la Nave con mayor puntuación.

Listing 12: Nave con el puntaje más alto

```
Nave con mayor numero de puntos: Fire
```

- Probando el método para desordenar todas las naves aleatoriamente y luego mostrarlas.

```
Nave 1:Mike
Posicion: 5D
Puntos: 65
Sigue con vida

Nave 2:Indio
Posicion: 2C
Puntos: 40
Fue destruido

Nave 3:Fire
Posicion: 4J
Puntos: 89
Sigue con vida

Nave 4:Delta
Posicion: 2A
Puntos: 43
Fue destruido

Nave 5:Whisky
Posicion: 1C
Puntos: 70
Sigue con vida
```

4.2. Actividad 2

- Para poder trabajar con arreglos de objetos en esta actividad y la siguiente, necesito crear una clase conveniente, en este caso, creé la clase 'Soldado' con sus atributos respectivos.

Listing 13: Clase Soldado

```
207 public class Soldado {
208     private String nombre;
209     private int vida;
210 
211     public void setNombre(String s) {
212         this.nombre = s;
213     }
214 
215     public void setVida(int n) {
216         this.vida = n;
217     }
218 
219     public String getNombre() {
220         return nombre;
221     }
222 
223     public int getVida() {
224         return vida;
225     }
226 }
```

- Ahora simplemente trabajé la actividad de leer los nombres y niveles de vida de cinco soldados, usando un arreglo de 'Soldados'.
- Para esto, almaceno los valores como atributos de los objetos Soldado, como la vida y el nombre.

Listing 14: Los 5 soldados

```
227 import java.util.*;
228 
229 public class VideoJuego {
230     public static void main(String[] args) {
231         Scanner sc = new Scanner(System.in);
232         Soldado[] army = new Soldado[5];
233         for (int i = 0; i < army.length; i++) {
234             army[i] = new Soldado();
235             System.out.println("Ingrese el nombre del soldado: ");
236             army[i].setNombre(sc.next());
237             System.out.println("Ingrese su nivel de vida");
238             army[i].setVida(sc.nextInt());
239         }
240         int i = 1;
241         for (Soldado s : army) {
242             System.out.println("Soldado " + i + " : " + s.getNombre());
243             System.out.println("Tiene " + s.getVida() + " puntos de vida\n");
244             i++;
245         }
246     }
247 }
```

- Y ahora trabajo el problema de los dos ejércitos usando arreglos de objetos, la lógica es similar pero para almacenar los nombres uso los atributos de los objetos.

Listing 15: Los 5 soldados

```
248 public class VideoJuego {
249     public static void main(String[] args) {
250         int len1 = (int) (Math.random() * 5 + 1);
251         int len2 = (int) (Math.random() * 5 + 1);
252         Soldado[] army1 = new Soldado[len1];
253         Soldado[] army2 = new Soldado[len2];
254         init(army1);
255         init(army2);
256         mostrar(army1, army2);
257         win(len1, len2);
258 
259     }
260 
261     public static void init(Soldado[] soldiers) {
262         for (int i = 0; i < soldiers.length; i++) {
263             soldiers[i] = new Soldado();
264             soldiers[i].setNombre("Soldado " + i);
265         }
266     }
267 
268     public static void win(int l1, int l2) {
269         if (l1 > l2) {
270             System.out.println("El ejercito 1 es ganador");
271         } else if (l1 == l2) {
272             System.out.println("Hay empate");
273         } else {
274             System.out.println("El ejercito 2 es ganador");
275         }
276     }
277 
278     public static void mostrar(Soldado[] army1, Soldado[] army2) {
279         System.out.println("Ejercito 1: ");
280         for (int i = 0; i < army1.length; i++) {
281             System.out.println(army1[i].getNombre());
282         }
283         System.out.println();
284         System.out.println("Ejercito 2: ");
285         for (int i = 0; i < army2.length; i++) {
286             System.out.println(army2[i].getNombre());
287         }
288 
289     }
290 }
```

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	2	
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 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		18	

6. Referencias

- https://www.w3schools.com/java/java_modifiers.asp
- https://puntocomnoesunlenguaje.blogspot.com/2013/04/llenar-un-array-con-numeros-aleatorios.html