

Informe de Laboratorio 01

Tema: Arreglos estándar

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
01	Arreglos estándar	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 11 Septiembre 2023	Al 17 Septiembre 2023

1. Tarea

1.1. Actividad 1

- Escribir un programa donde se creen 5 soldados considerando sólo su nombre. Ingresar sus datos y después mostrarlos.
- Restricción: se realizará considerando sólo los conocimientos que se tienen de FP1 y sin utilizar arreglos estándar, sólo usar variables simples.

1.2. Actividad 2

- Escribir un programa donde se creen 5 soldados considerando su nombre y nivel de vida. Ingresar sus datos y después mostrarlos.
- Restricción: se realizará considerando sólo los conocimientos que se tienen de FP1 y sin utilizar arreglos estándar, sólo usar variables simples.

1.3. Actividad 3

- Escribir un programa donde se creen 5 soldados considerando su nombre y nivel de vida. Ingresar sus datos y después mostrarlos.
- Restricción: aplicar arreglos estándar.

1.4. Actividad 4

- Escribir un programa donde se creen 5 soldados considerando su nombre y nivel de vida. Ingresar sus datos y después mostrarlos.
- Restricción: aplicar arreglos estándar.

1.5. Actividad 5

- Escribir un programa donde se creen 2 ejércitos, cada uno con un número aleatorio de soldados entre 1 y 5, considerando sólo su nombre. Sus datos se inicializan automáticamente con nombres tales como “Soldado0”, “Soldado1”, etc. Luego de crear los 2 ejércitos se deben mostrar los datos de todos los soldados de ambos ejércitos e indicar qué ejército fue el ganador.
- Restricción: aplicar arreglos estándar y métodos para inicializar los ejércitos, mostrar ejército y mostrar ejército ganador. La métrica a aplicar para indicar el ganador es el mayor número de soldados de cada ejército, puede haber empates. (Todavía no aplicar arreglo de objetos)

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Home Single Language 64 bits 22H2.2283
- 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.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/RyanValdivia/fp2-23b.git>
- URL para el laboratorio 01 en el Repositorio GitHub.
- <https://github.com/RyanValdivia/fp2-23b/tree/main/fase01/lab01>

4. Actividades

4.1. Actividad 1

- Se realizó un commit con la versión final del código de la Actividad 1

Listing 1: Comentando la versión final de esta actividad

```
$ git log VideoJuego.java
commit c161ae0edeee853c558913d952eaaa2320b2ae0a
Author: RYAN VALDIVIA <rvaldiviase@unsa.edu.pe>
Date: Fri Sep 15 09:20:15 2023 -0500
```

Actividad 1: Creando un programa que cree 5 soldados, ingrese sus nombres y luego los muestre

- Conteniendo el siguiente código:

Listing 2: Los 5 soldados

```
6  import java.util.*;
7
8  public class VideoJuego {
9      public static void main(String[] args) {
10         Scanner sc = new Scanner(System.in);
11
12         System.out.println("Ingrese los nombres de los soldados: ");
13         String sold1 = sc.next();
14         String sold2 = sc.next();
15         String sold3 = sc.next();
16         String sold4 = sc.next();
17         String sold5 = sc.next();
18
19         System.out.println("Soldado 1: " + sold1);
20         System.out.println("Soldado 2: " + sold2);
21         System.out.println("Soldado 3: " + sold3);
22         System.out.println("Soldado 4: " + sold4);
23         System.out.println("Soldado 5: " + sold5);
24
25         sc.close();
26     }
27 }
```

- Debido a la restricción de no usar arreglos, declaré cinco variables para contener a los nombres de los cinco soldados, para posteriormente mostrarlos en consola.

Listing 3: Compilación y ejecución del programa

```
$ javac VideoJuego.java
$ java VideoJuego
Ingrese los nombres de los soldados:
Juan Carlos Pepe Marco Antonio
Soldado 1: Juan
Soldado 2: Carlos
Soldado 3: Pepe
Soldado 4: Marco
Soldado 5: Antonio
```

4.2. Actividad 2

- Ahora, se nos añade la tarea de almacenar los nombres y los niveles de vida de 5 soldados, sin usar arreglos.
- Para esto, simplemente declaré y leí 5 variables más, que contengan los niveles de vida de los soldados para mostrarlos después

Listing 4: Los 5 soldados + Los 5 niveles de vida

```
37 String sold1 = sc.next();
38 int lvl1 = sc.nextInt();
39 String sold2 = sc.next();
40 int lvl2 = sc.nextInt();
41 String sold3 = sc.next();
42 int lvl3 = sc.nextInt();
43 String sold4 = sc.next();
44 int lvl4 = sc.nextInt();
45 String sold5 = sc.next();
46 int lvl5 = sc.nextInt();
47
48 System.out.println("Soldado 1: " + sold1 + " y su nivel de vida es: " + lvl1);
49 System.out.println("Soldado 2: " + sold2 + " y su nivel de vida es: " + lvl2);
50 System.out.println("Soldado 3: " + sold3 + " y su nivel de vida es: " + lvl3);
51 System.out.println("Soldado 4: " + sold4 + " y su nivel de vida es: " + lvl4);
52 System.out.println("Soldado 5: " + sold5 + " y su nivel de vida es: " + lvl5);
```

Listing 5: Ejecutando y nombrando soldados

```
$ javac VideoJuego.java
$ java VideoJuego
Ingrese los nombres de los soldados y su respectivo nivel de vida:
Juan 12 Carlos 16 Pepe 09 Marco 11 Antonio 10
Soldado 1: Juan y su nivel de vida es: 12
Soldado 2: Carlos y su nivel de vida es: 16
Soldado 3: Pepe y su nivel de vida es: 9
Soldado 4: Marco y su nivel de vida es: 11
Soldado 5: Antonio y su nivel de vida es: 10
```

4.3. Actividad 3

- Para este ejercicio, ya tenemos "desbloqueada" la utilización de arreglos estándar, esto facilitará mucho la tarea de almacenar los valores.

Listing 6: Los 5 soldados, pero con arreglos

```
63 import java.util.*;
64 public class VideoJuego {
65     public static void main(String[] args) {
66         Scanner sc = new Scanner(System.in);
67         System.out.println("Ingrese los nombres de los soldados");
68         String[] soldiers = new String[5];
69
70         for (int i = 0; i < 5; i++) {
71             soldiers[i] = sc.next();
72         }
73         for (int i = 0; i < 5; i++) {
74             System.out.println("Soldado " + i + ": " + soldiers[i]);
75         }
76         sc.close();
77     }
78 }
```

- La lógica para este código, fue declarar un arreglo de Strings para contener los nombres de los soldados, y hacer un ciclo for para llenarlo.
- Posteriormente, usé un ciclo for para imprimir en consola los datos previamente almacenados.

Listing 7: Probando con arreglos

```
$ javac VideoJuego.java
$ java VideoJuego
Ingrese los nombres de los soldados:
Juan Carlos Pepe Marco Antonio
Soldado 0: Juan
Soldado 1: Carlos
Soldado 2: Pepe
Soldado 3: Marco
Soldado 4: Antonio
```

- Es el mismo resultado, pero programado de una forma más práctica.

4.4. Actividad 4

- En esta actividad, la lógica es la misma, solo creando dos arreglos para contener los nombres y los niveles de vida de los soldados.

Listing 8: Usando arreglos para los niveles de vida y nombres

```
88 import java.util.*;
89
90 public class VideoJuego {
91     public static void main(String[] args) {
92         Scanner sc = new Scanner(System.in);
93
94         System.out.println("Ingrese los nombres de los soldados y sus respectivos
95                             niveles de vida");
96
97         String[] soldiers = new String[5];
98         int[] levels = new int[5];
99
100        for (int i = 0; i < 5; i++) {
101            soldiers[i] = sc.next();
102            levels[i] = sc.nextInt();
103        }
104
105        for (int i = 0; i < 5; i++) {
106            System.out.println("Soldado " + i + ": " + soldiers[i] + " y su nivel de
107                                vida es: " + levels[i]);
108        }
109        sc.close();
110    }
111 }
```

4.5. Actividad 5

- Esta actividad ocupa algo más elaborado, realizando múltiples métodos para hacer la tarea más sencilla.
- Primero, creé un método que llene un arreglo de strings con "Soldado- su respectivo número, dependiendo de su longitud, usando un ciclo for.

Listing 9: Método para inicializar los arreglos de soldados

```
111 public static void init(String[] strs) {  
112     for (int i = 0; i < strs.length; i++) {  
113         strs[i] = "Soldado" + i;  
114     }  
115 }
```

- Posteriormente, creé otro método que, dados dos ejércitos, determine cual es el ganador, basándose en la mayor cantidad de soldados.

Listing 10: Método para declarar al ganador

```
116 public static void win(int l1, int l2) {  
117     if (l1 > l2) {  
118         System.out.println("El ejercito 1 es ganador");  
119     } else if (l1 == l2) {  
120         System.out.println("Hay empate");  
121     } else {  
122         System.out.println("El ejercito 2 es ganador");  
123     }  
124 }
```

- Además, implementé un método para imprimir ambos ejércitos una vez creados.

Listing 11: Método para imprimir ambos arreglos(ejércitos)

```
125 public static void mostrar(String[] strs1, String[] strs2) {  
126     System.out.println("Ejercito 1: ");  
127     for (String s : strs1) {  
128         System.out.println(s);  
129     }  
130     System.out.println();  
131  
132     System.out.println("Ejercito 2: ");  
133     for (String s : strs2) {  
134         System.out.println(s);  
135     }  
136     System.out.println();  
137 }
```

- Finalmente, solo quedaba armar todo en el método main, obteniendo dos tamaños de ejército aleatorios y declarando los dos arreglos que contendrán a los ejércitos.

Listing 12: Código final armado

```
138 public static void main(String[] args) {  
139     int len1 = (int) (Math.random() * 5 + 1);  
140     int len2 = (int) (Math.random() * 5 + 1);  
141     String[] army1 = new String[len1];  
142     String[] army2 = new String[len2];  
143     init(army1);  
144     init(army2);  
145     mostrar(army1, army2);  
146     win(len1, len2);  
147 }
```

Listing 13: Ejecutando el Video juego

```
$ javac VideoJuego.java  
$ java VideoJuego  
Ejercito 1:  
Soldado0  
Soldado1  
Soldado2  
Soldado3  
Soldado4  
  
Ejercito 2:  
Soldado0  
Soldado1  
Soldado2  
Soldado3  
Soldado4  
  
Hay empate
```

Listing 14: Probando de nuevo, debería ser aleatorio

```
$ javac VideoJuego.java  
$ java VideoJuego  
Ejercito 1:  
Soldado0  
Soldado1  
Soldado2  
  
Ejercito 2:  
Soldado0  
Soldado1  
Soldado2  
Soldado3  
  
El Ejercito 2 es ganador
```

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 cumple 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 los ítems.
- 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 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	3	
Total		20		17	

6. Referencias