Informe de Laboratorio 01
Tema: Arreglos Estándar

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
01	Arreglos Estándar	02 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 18 Septiembre 2023	Al 20 Septiembre 2023

1. Tarea

- 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.
- 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.
- Escribir un programa donde se creen 5 soldados considerando sólo su nombre. Ingresar sus datos y después mostrarlos. Restricción: aplicar arreglos estándar.
- 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. (Todavía no aplicar arreglo de objetos)
- 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)
- Utilizar Git para evidenciar su trabajo.
- Enviar trabajo al profesor en un repositorio GitHub Privado, dándole permisos como colaborador.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Pro 22H2 64 bits.
- VIM 9.0.
- OpenJDK 64-Bits 17.0.7.
- Git 2.41.1.
- Cuenta en GitHub con el correo institucional.
- Variables Simples
- Arreglos Estándar

3. URL de Repositorio Github

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

4. Actividades con el repositorio GitHub

4.1. Creando e inicializando repositorio GitHub

- Como es el primer laboratorio se creo el repositorio GitHub.
- Se realizaron los siguientes comandos en la computadora:

Listing 1: Creando directorio de trabajo

```
$ mkdir e:/fp2-23b/
```

Listing 2: Dirijíéndonos al directorio de trabajo

```
$ cd e:/fp2-23b/
```

Listing 3: Creando directorio para la primera fase

```
$ mkdir e:/fp2-23b/fase01/
```

Listing 4: Inicializando directorio para repositorio GitHub

```
$ git init
$ git config --global user.name "Hernan Andy Choquehuanca Zapana"
$ git config --global user.email hchoquehuancaz@unsa.edu.pe
$ git branch -M main
$ git remote add origin https://github.com/hernanchoquehuanca/fp2-23b.git
$ git add Videojuego.java
$ git commit -m "Agregando el saludo a VideoJuego.java con un mensaje de Bienvenida"
$ git push -u origin main
```

4.2. Commits

4.2.1. 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.

Listing 5: Primer commit creando variables para los nombres de soldados

```
$ mkdir lab01
$ touch lab01/Ejercicio01.java
$ git add .
$ git commit -m "Ejercicio01 realizado con variables simples (Strings) para los datos
    de los 5 soldados"
$ git push -u origin main
```

- En el segundo commit se le agregó un mensaje al recibir e imprimir los nombres
- El código fue el siguiente:

Listing 6: Ejercicio01.java

```
// Laboratorio Nro 01 - Ejercicio01
// Autor : Hernan Andy
// Colaboro : -

import java.util.Scanner;
class Ejercicio01 {
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    System.out.print("Ingrese el primer soldado: ");
    String soldier1 = sc.next();
    System.out.print("Ingrese el segundo soldado: ");
    String soldier2 = sc.next();
    System.out.print("Ingrese el tercer soldado: ");
    String soldier3 = sc.next();
    System.out.print("Ingrese el cuarto soldado: ");
    String soldier4 = sc.next();
    System.out.print("Ingrese el quinto soldado: ");
    String soldier5 = sc.next();

    System.out.println("Lista de soldados: ");
    System.out.println("Soldado Nro1 " + soldier1);
    System.out.println("Soldado Nro2 " + soldier2);
    System.out.println("Soldado Nro3 " + soldier3);
    System.out.println("Soldado Nro4 " + soldier4);
    System.out.println("Soldado Nro5 " + soldier5);
  }
}
```

4.2.2. 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.

Listing 7: Ejercicio02.java

```java
// Laboratorio Nro 01 - Ejercicio02
// Autor : Hernan Andy
// Colabor : -
// Tiempo : -

import java.util.Scanner;
class Ejercicio02 {
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);

    System.out.print("Ingrese el nombre del primer soldado: ");
    String soldier1 = sc.next();
    System.out.print("Vida: ");
    int healt1 = sc.nextInt();
    System.out.print("Ingrese el segundo soldado: ");
    String soldier2 = sc.next();
    System.out.print("Vida: ");
    int healt2 = sc.nextInt();
    System.out.print("Ingrese el tercer soldado: ");
    String soldier3 = sc.next();
    System.out.print("Vida: ");
    int healt3 = sc.nextInt();
    System.out.print("Ingrese el cuarto soldado: ");
    String soldier4 = sc.next();
    System.out.print("Vida: ");
    int healt4 = sc.nextInt();
    System.out.print("Ingrese el quinto soldado: ");
    String soldier5 = sc.next();
    System.out.print("Vida: ");
    int healt5 = sc.nextInt();

    System.out.println("Lista de soldados: ");
    System.out.println("Soldado Nro1 : " + soldier1 + " - vida :" + healt1);
    System.out.println("Soldado Nro2 : " + soldier2 + " - vida :" + healt2);
    System.out.println("Soldado Nro3 : " + soldier3 + " - vida :" + healt3);
    System.out.println("Soldado Nro4 : " + soldier4 + " - vida :" + healt4);
    System.out.println("Soldado Nro5 : " + soldier5 + " - vida :" + healt5);
  }
}
```

Listing 8: Commit: Ejercicio02 realizado con variables simples (String e int) tanto para los nombres, como para la vida

```
$ git add .
$ git commit -m "Ejercicio02 realizado con variables simples (String e int) tanto para
    los nombres, como para la vida"
$ git push -u origin main
```

- Utilizando el código Ejercicio01.java, se agregó variables enteras (int) para la vida de los soldados

4.2.3. Actividad 3 : Escribir un programa donde se creen 5 soldados considerando sólo su nombre. Ingresar sus datos y después mostrarlos.

Restricción: Aplicar arreglos estándar.

- Utilizando arreglos estándar y un bucle for para recibir los datos y asignarles una posición en el arreglo "soldiers"según el índice i"

Listing 9: Ejercicio03.java

```java
// Laboratorio Nro 01 - Ejercicio03
// Autor : Hernan Andy
// Colabor : -
// Tiempo : -

import java.util.Scanner;
class Ejercicio03 {
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    String[] soldiers = new String[5];

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


  }
}
```

Listing 10: Commit: Ejercicio03 ahora usando arrays, utilizando un bucle for para recibir los nombres

```
$ git add .
$ git commit -m "Ejercicio03 ahora usando arrays, utilizando un bucle for para recibir
    los nombres"
$ git push -u origin main
```

- Para finalizar con ayuda de otro bucle for se imprimirán los nombres de los soldados ingresados

Listing 11: Ejercicio03.java

```java

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

Listing 12: Commit: Agregando un segundo bucle for para mostrar los datos (nombres)

```
$ git add .
$ git commit -m "Agregando un segundo bucle for para mostrar los datos (nombres)"
$ git push -u origin main
```

- El código con los cambios ya mencionados es el siguiente:

Listing 13: Ejercicio03.java

```java
// Laboratorio Nro 01 - Ejercicio03
// Autor : Hernan Andy
// Colabor : -
// Tiempo : -

import java.util.Scanner;
class Ejercicio03 {
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    String[] soldiers = new String[5];

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

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

  }
}
```

4.2.4. 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. (Todavía no aplicar arreglo de objetos)

- Utilizando el código de Ejercicio03.java
- Se agregó un bucle for donde se recibiría la vida de los soldados en un arreglo llamado "healt"

Listing 14: Ejercicio04.java

```java
    int[] healt = new int[5];

    for (int i = 0; i < healt.length; i++) {
      System.out.print("Ingrese la vida del soldado Nro " + (i + 1) + " : ");
      healt[i] = sc.nextInt();
    }
```

Listing 15: Commit: Ejercicio04 usando los bucles del ejercicio anterior para los nombres y creando un bucle para recibir las vidas

```
$ git add .
$ git commit -m "Ejercicio04 usando los bucles del ejercicio anterior para los nombres
    y creando un bucle para recibir las vidas"
$ git push -u origin main
```

- Finalmente se cambió la entrada de la vida de los soldados, colocándolos dentro del mismo bucle for que recibe los nombres

Listing 16: Ejercicio04.java

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

      System.out.print("Vida: ");
      healt[i] = sc.nextInt();
    }
```

Listing 17: Commit: Recibiendo la vida de los soldados dentro del mismo bucle que los nombres para reducir código

```
    $ git add .
    $ git commit -m "Recibiendo la vida de los soldados dentro del mismo bucle que los
        nombres para reducir cdigo"
    $ git push -u origin main
```

- El código con los cambios ya mencionados es el siguiente:

Listing 18: Ejercicio04.java

```
// Laboratorio Nro 01 - Ejercicio04
// Autor : Hernan Andy
//  Colabor  : -
// Tiempo : -

import java.util.Scanner;
class Ejercicio04 {
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);

    String[] soldiers = new String[5];
    int[] healt = new int[5];

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

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

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

  }
}
```

4.2.5. 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)

- Comenzando por el método que creará los ejércitos, este generará un entero n que indicará el número de soldados del ejército

Listing 19: Ejercicio05.java

```
public static String[] createArmy() {
  int n = (int) (Math.random() * 5) + 1;
  String [] army = new String[n];
  for (int i = 0; i < n; i++) {
    army[i] = "Soldado" + i;
  }
  return army;
}
```

- Seguidamente implementamos el método para mostrar los ejércitos que recibe como parámetro un arreglo de Strings

Listing 20: Ejercicio05.java

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

- Ahora añadimos un método que mostrará al ejército ganador, basándose en el tamaño del arreglo, ya que de esa manera se evalúa el ejército más grande
- En caso de que ambos ejércitos tengan el mismo tamaño se dará un empate, esto con la ayuda de condicionales (if - else if)

Listing 21: Ejercicio05.java

```
public static void armyWinner(String[] army1, String[] 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");
  }
}
```

- Finalmente ahora que ya tenemos los métodos, los usaremos en el main tal como se muestra a continuación :

Listing 22: Ejercicio05.java

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

  System.out.println("Ejercito Nro 1");
  showArmy(army1);
  System.out.println("\nEjercito Nro 2" );
  showArmy(army2);

  armyWinner(army1, army2);
}
```

- En el último commit se subió todos los métodos unidos y siendo utilizandos en el main para darle funcionalidad al programa
- CADA UNO DE LOS MÉTODOS PRESENTADOS TIENEN SUS RESPECTIVOS COMMITS

Listing 23: Commit: Agregando el main utilizando todos los metodos elaborados para terminar con el programa

```
    $ git add .
    $ git commit -m "Agregando el main utilizando todos los metodos elaborados para
        terminar con el programa"
    $ git push -u origin main
```

- El código con todos los cambios ya mencionados es el siguiente:

Listing 24: Ejercicio05.java

```java
// Laboratorio Nro 01 - Ejercicio05
// Autor : Hernan Andy
// Colabor : -
// Tiempo : -

class Ejercicio05 {
  public static void main(String[] args) {
    String[] army1 = createArmy();
    String[] army2 = createArmy();

    System.out.println("Ejercito Nro 1");
    showArmy(army1);
    System.out.println("\nEjercito Nro 2" );
    showArmy(army2);

    armyWinner(army1, army2);
  }

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

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

    return army;
  }

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

  public static void armyWinner(String[] army1, String[] 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");
    }
  }
}
```

4.3. Estructura de laboratorio 01

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

```
lab01
|---|--- Ejercicio01.java
|---|--- Ejercicio02.java
|---|--- Ejercicio03.java
|---|--- Ejercicio04.java
|---|--- Ejercicio05.java
|---|--- VideoJuego.java
|--- latex
   |--- img
   |   |--- logo_abet.png
   |   |--- logo_episunsa.png
   |   |--- logo_unsa.jpg
   |--- Informe_lab01.pdf
   |--- Informe_lab01.tex
   |--- src
       |---|--- Ejercicio01v2.java
       |---|--- Ejercicio02.java
       |---|--- Ejercicio03v1.java
       |---|--- Ejercicio03v2.1.java
       |---|--- Ejercicio03v2.java
       |---|--- Ejercicio04.java
       |---|--- Ejercicio04v1.java
       |---|--- Ejercicio04v2.java
       |---|--- Ejercicio05.java
       |---|--- Ejercicio05v1.java
       |---|--- Ejercicio05v2.java
       |---|--- Ejercicio05v3.java
       |---|--- Ejercicio05v4.java
       |---|--- VideoJuego.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