Informe de Laboratorio 01

Tema: Arreglos Estandar

Nota

Estudiante	Escuela	Asignatura
Jhonatan David Arias Quispe jariasq@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de Programacion 2 Semestre: II Código: 1701213

Laboratorio	Tema	Duración
01	Arreglos Estandar	04 horas

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

1. Tarea

- 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.
- 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.
- Actividad 3: escribir un programa donde se creen 5 soldados considerando sólo su nombre. Ingresar sus datos y después mostrarlos. Restricción: aplicar arreglos estándar.
- 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)
- 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 ArchCraft GNU Linux 64 bits Kernell
- NeoVim
- OpenJDK 64-Bit 20.0.1
- Git 2.42.0
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.
- Creacion de programas con CLI

3. URL de Repositorio Github

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

4. Actividad 1: 5 soldados con nombre

4.1. Creando una clase Soldado

- Necesitamos una clase Soldado que tenga como atributo Name
- Para ello definimos la clase en java:

Listing 1: Clase Soldado

```java
public class Soldado{
  public String name;

  public void setName(String str){
    this.name = str;
  }
  public String getName(){
    return this.name;
  }
}
```

4.2. Creando el archivo VideoJuego

- Ahora necesitamos establecer sus nombres y mostrarlos en pantalla
- Para eso creamos el archivo principal VideoJuego

Listing 2: Creando el archivo VideoJuego.java

```
public class VideoJuego{
  public static void main(String[] args){
    Soldado rambo = new Soldado("Rambo");
    Soldado goomp = new Soldado("Goomp");
    Soldado joel = new Soldado("Joel");
    Soldado mauricio = new Soldado("Mauricio");
    Soldado rogelio = new Soldado("Rogelio");

    System.out.println("Escuadron Sistemas:\n"
    + rambo.getName() + "\n"
    + goomp.getName() + "\n"
    + joel.getName() + "\n"
    + mauricio.getName() + "\n"
    + rogelio.getName() + "\n");
  }
}
```

4.3. Commits

Listing 3: Primer Commit Creando carpeta/archivo para laboratorio 01

```
commit 6f8fdd7000a3c75fb8409f05e12df7e941cfb387
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Tue Sep 19 13:20:37 2023 -0500

    Actividad 1: Cinco soldados con nombres

commit 6f1f099db7c6a8a7e490f9e58a9eb7366681136f
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Sep 18 17:50:42 2023 -0500

    Avance Actividad 1

commit 3b6c3c402b581ca43216641e6ab811dad6007017
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Sep 18 08:23:46 2023 -0500

    Actividad 1: inicializando archivos
```

- Registro de los commits con **git log**
- En total fueron 3 commits

5. Actividad 2: Agregando el nivel de vida

- Necesitamos agregar otro atributo: life
- El item sera de tipo entero para facilitar futuras implementaciones
- Adicionalmente necesitaremos configurar los Getters y Setters para los atributos

5.1. Agregando atributos y metodos getters y setters

Listing 4: Creando un nuevo atributo para Soldado

```
public int life;
public void setLife(int life){
  this.life = life;
}

//SECCION DE GETERS

public String getName(){
  return this.name;
}

public int getLife(){
  return this.life;
}
```

5.2. Ejecucion

Listing 5: Ejecucion del codigo

```
javac VideoJuego.java
java VideoJuego

Escuadron Sistemas:
Rambo
Goomp
Joel
Mauricio
Rogelio
```

6. Actividad 3: Permitir al usuario definir los datos

6.1. Entrada del usuario

- Como no podemos usar arreglos de objetos, usaremos un arreglo de String para almacenar los datos

Listing 6: Almacenando los nombres usando un ciclo for

```
String[] names = new String[5];
for(int i = 0; i < 5; i++){
  System.out.println("\nIngrese el nombre del soldado numero " + (i + 1) + ":");
  names[i] = sc.next();
}
```

6.2. Ejecucion

Listing 7: Salida por consola

```
Ingrese el nombre del soldado numero 1:
Romel

Ingrese el nombre del soldado numero 2:
Reginald

Ingrese el nombre del soldado numero 3:
Ricardo

Ingrese el nombre del soldado numero 4:
Rambo

Ingrese el nombre del soldado numero 5:
Rinosio
```

- Como primera parte, el juego te pide ingresar el nombre de los soldados

Listing 8: Muestreo de datos al usuario

```
=====DATOS DE SOLDADOS=====
Soldado: Romel
Nivel de vida: 0

Soldado: Reginald
Nivel de vida: 0

Soldado: Ricardo
Nivel de vida: 0

Soldado: Rambo
Nivel de vida: 0

Soldado: Rinosio
Nivel de vida: 0
```

- Como no hemos ingresado el nivel de vida de los soldados, es 0

7. Actividad 4: Digitando el nombre y nivel de vida

- Unicamente agregamos otro array que represente el nivel de vida

Listing 9: Pidiendo al usuario la vida de los soldados

```
String[] names = new String[5];
int[] lifes = new int[5];

for(int i = 0; i < 5; i++){
  System.out.println("\nIngrese el nombre del soldado numero " + (i + 1) + ":");
  names[i] = sc.next();

  System.out.println("Ingrese el nivel de vida del soldado numero " + (i + 1) + ":");
```

```
        lifes[i] = sc.nextInt();
    }
```

- Ahora agregamos la posibilidad de ver estos datos

Listing 10: Codigo para salida de datos

```
    System.out.println("\n=====DATOS DE SOLDADOS=====");
    System.out.println("Soldado: " + s1.getName() + " \nNivel de vida: " + s1.getLife() +
        "\n");
    System.out.println("Soldado: " + s2.getName() + " \nNivel de vida: " + s2.getLife() +
        "\n");
    System.out.println("Soldado: " + s3.getName() + " \nNivel de vida: " + s3.getLife() +
        "\n");
    System.out.println("Soldado: " + s4.getName() + " \nNivel de vida: " + s4.getLife() +
        "\n");
    System.out.println("Soldado: " + s5.getName() + " \nNivel de vida: " + s5.getLife() +
        "\n");
```

7.1. Ejecucion en consola

Listing 11: Porcion de ejecicion de VideoJuego.java

```
    Ingrese el nombre del soldado numero 1:
    Ronaldo
    Ingrese el nivel de vida del soldado numero 1:
    12

    Ingrese el nombre del soldado numero 3:
    Rias
    Ingrese el nivel de vida del soldado numero 3:
    100

    Ingrese el nombre del soldado numero 4:
    Ryan
    Ingrese el nivel de vida del soldado numero 4:
    12

    =====DATOS DE SOLDADOS=====
    Soldado: Ronaldo
    Nivel de vida: 12

    Soldado: Rias
    Nivel de vida: 100

    Soldado: Ryan
    Nivel de vida: 12
```

8. Actividad 5: Simulacion de batalla entre dos ejercitos

- El codigo esta estructurado en metodos, y a cada uno le pertenece un commit

8.1. Commit 1: Metodo de inicializacion de ejercitos

Listing 12: Metodo initializeArmy()

```
public static String[] initializeArmy(){
  Random rand = new Random();
  int randNum = rand.nextInt(5) + 1;
  String[] army = new String[randNum];

  for(int i = 0; i < randNum; i++){
    army[i] = "Soldado " + (i + 1);
  }
  return army;
}
```

- Usamos la clase Random para generar un array con un numero al azar
- Luego, con un bucle for recorremos el arreglo estableciendo el nombre de los soldados

8.2. Commit 2: Creacion del metodo display

- Necesitamos un metodo para mostrar la cantidad de soldados que tiene un ejercito

Listing 13: Metodo display()

```
public static void displayArmy(String[] army){
  System.out.println("\n===== Army Soldiers =====");
  for(String soldier : army){
    System.out.println(" " + soldier);
  }
}
```

- Usamos un bucle for : each para imprimir uno a uno los nombres de los soldados

8.3. Commit 3: Implementacion de un metodo para determinar al ganador

- Para esto necesitamos saber el numero exacto de soldados en cada ejercito:

Listing 14: Metodo whoWins()

```
public static String whoWins(String[] army1, String[] army2){
  if (army1.length > army2.length)
    return "\n***** Army 1 is the winner! *****";

  if (army2.length > army1.length)
    return "\n***** Army 2 is the winner! *****";

  return "\n***** It's a tie. No clear winner. *****";
}
```

- Lo hacemos usando el metodo .length en nuestros arrays
- Note la forma peculiar de usar los condicionales if, al ser un metodo podemos omitir los 'else'

8.4. Commit 4: Implementacion del metodo main y CLI

8.4.1. Banner de juego

- Para dar la experiencia de juego interactivo, hacemos un banner de inicio:

Listing 15: Banner del juego

```
                    Welcome to the Battle
                       Simulator Game!

                ***** Prepare for battle! *****
```

- Este arte/estilo de hacer programas por consola se denomina CLI
- CLI quiere decir Çommand Line Interface", Ïnterfaz de Linea de Comandos", y en lo personal me gusta mucho

8.4.2. Implementacion del main

- Necesitamos inicializar dos ejercitos, mostrarlos, e imprimir cual de los dos resulto ganador

Listing 16: Metodo main() de VideoJuego.java

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

      System.out.println("                                          ")
      System.out.println("    Welcome to the Battle    ");
      System.out.println("       Simulator Game!       ");
      System.out.println("                                          ")

      System.out.println("\n***** Prepare for battle! *****");

      displayArmy(army1);
      displayArmy(army2);

      System.out.println(whoWins(army1, army2));

    }
```

- Hacemos uso de los metodos anteriormente creados
- Ahora pasamos a hacer las pruebas para la optimizacion

8.5. Ejecucion

- Para nuestro caso de simulacion unicamente tenemos que compilar y ejecutar:

Listing 17: Ejecucion en la linea de comandos

```
javac VideoJuego.java
```

```
java VideoJuego

      Welcome to the Battle
         Simulator Game!


***** Prepare for battle! *****

===== Army Soldiers =====
 Soldado 1
 Soldado 2
 Soldado 3
 Soldado 4

===== Army Soldiers =====
 Soldado 1
 Soldado 2

***** Army 1 is the winner! *****
```

- Como hemos podido ver, ha ganado el ejercito 1 ya que tiene 2 soldados de mas
- Con esto hemos finalizado todas las actividades

9. Rúbricas

9.1. 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.

9.2. 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.
- 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	4	
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	1.5	
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	1.5	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	1.5	
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		16.5	