Informe de Laboratorio 01

Tema: Arreglos Estandar

Nota

Estudiante	Escuela	Asignatura
Julio Rubén Chura Acabana jchuraaca@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	F. de Programción 2 Semestre: I Código: 20230472

Laboratorio	Tema	Duración
01	Arreglos Estandar	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 11 Septiembre 2023	Al 13 Septiembre 2023

1. Tarea

- Escribir un código donde se creen 5 soldados considerando su nombre usando variables simples
- Crear 5 soldados con sus respectivos nombres y nivel de vida usando variables simples
- Crear 5 soldados considerando su nombre usando arreglos estandar.
- Crear 5 soldados considerando su nombre y nivel de vida usando arreglos
- Crear dos arreglos de soldados y determinar quien gana considerando como condición de victoria el bando que tenga mayor cantidad de soldados
- Trabajar los 5 ejercicios en un mismo archivo pero se deberá subir al repositorio cada versión. Luego se presentará un informe de la sesión

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows
- VIM 9.0.
- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Arreglos Estándar

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JulioChura/fp2-23b.git`
- URL para el laboratorio 01 en el Repositorio GitHub.
- `https://github.com/JulioChura/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 fp2-23b\fase01\lab01
cd fp2-23b
mkdir fase02
mkdir fase03
```

Listing 2: Dirijiéndonos al directorio de trabajo

```
cd fp2-23b
```

Listing 3: Inicializando directorio para repositorio GitHub

```
cd C:\Users\USUARIO\Desktop\fp2-23b
git init
git config --global user.name "Julio Chura"
git config --global user.email jchuraaca@unsa.edu.pe
git add README.md
git commit -m "Se agrega el proyecto"
git branch -M main
git remote add origin https://github.com/JulioChura/fp2-23b.git
git push -u origin main
```

4.2. Commits

Listing 4: Primer Commit Creando carpeta/archivo para laboratorio 01

```
cd fase01/lab01
vim VideoJuego.java
git add .
git commit -m "Creando el archivo VideoJuego.java"
$ git push -u origin main
```

- Se creo el archivo **.gitignore** para no considerar los archivos ***.class** que son innecesarios hacer seguimiento.

Listing 5: Creando .gitignore

```
cd C:\Users\USUARIO\Desktop\fp2-23b
vim .gitignore
```

Listing 6: lab01/.gitignore

```
*.class
```

Listing 7: Commit: Creando .gitignore para archivos *.class y creando VideoJuego.java

```
git add .
git commit -m "Subiendo gitignore"
git push -u origin main
cd fp2-23b\fase01\lab01
vim VideoJuego.java
```

- Para el siguiente commit, se resuelve el primer y segundo ejercicio de la práctica, el cual consiste en crear 5 soldados con su respectivo nombre y con su nivel de vida, para ello se hace uso de variables simples como se indica
- En esta parte, por lo general las líneas 12, 13, 14, 15 del código se repetirán de forma constante teniendo pequeñas variaciones. Más antes de esta versión se realizó otra pero ingresando solo los nombres de los soldados, debido a que esta versión abarca la primera parte se optó por considerar esta

```
 6 public class VideoJuego{
 7     public static void main(String[] args){
 8         Scanner sc = new Scanner(System.in);
 9         String soldier1, soldier2, soldier3, soldier4, soldier5;
10         int life1, life2, life3, life4, life5;
11 
12         System.out.println("Ingrese el nombre del soldado 1");
13         soldier1 = sc.next();
14         System.out.println("Ingrese el nivel de vida del soldado 1");
15         life1 = sc.nextInt();
```

Listing 8: Subiendo al repositorio los cambios realizados al código VideoJuego.java

```
git add VideoJuego.java
git commit -"Poner nivel de vida de cada soldado y mostrarlo"
git push -u origin main
```

Listing 9: Uso de arreglos y números aleatorios para crear los soldados y su vida

```
vim VideoJuego.java
```

```
public class VideoJuego{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);

        String[] army = new String[5];
        for(int i = 0; i < army.length; i++){
            System.out.println("Enter the name of soldier "+ (i+1));
            army[i] = sc.next();
        }
        for(int i = 0; i < army.length; i++){
            System.out.println("Soldier "+(i+1)+": "+ army[i]);
            System.out.println("Life: "+ life()+"\t");
        }

    }
    public static int life(){
        Random lifePoints = new Random();
        return lifePoints.nextInt(10)+1;
    }
}
```

Listing 10: Subiendo los cambios al repositorio

```
git add VideoJuego.java
git commit -m "Anadiendo la vida de cada soldado usando numeros aleatorios"
```

Listing 11: Ejercicio 5: Enfrentamiento entre dos ejercitos

```
vim VideoJuego.java
```

Listing 12: VideoJuego.java terminado

```
1  // Laboratorio Nro 1 - Ejercicio 5
2  // Autor: Julio
3  // Tiempo: 20 minutos
4  // No hubo colaboradores
5  import java.util.*;
6  public class VideoJuego{
7    public static void main(String[] args){
8      Scanner sc = new Scanner(System.in);
9    String[] army1 = generateArmy();
10   String[] army2 = generateArmy();
11
12   System.out.println("ARMY 1");
13   printArray(army1);
```

```
   System.out.println("ARMY 2");
   printArray(army2);

   if (army1.length < army2.length ) {
     System.out.println("The winner is the army 2");
     } else if ( army1.length > army2.length) {
       System.out.println("The winner is the army 1");
     } else {
      System.out.println("IT WAS A TIE");
     }
    }

    public static String[] generateArmy(){
      Random random = new Random();
     int amount = random.nextInt(5)+1;
     String[] army = new String[amount];
     for(int i = 0; i < army.length; i++){
     army[i] = "Soldier"+(i+1);
     }
     return army;
    }

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

Listing 13: Compilando y probando código

```
    javac VideoJuego.java
    java VideoJuego
    ARMY 1
    Soldier1
    Soldier2
    Soldier3

    ARMY 2
    Soldier1

    The winner is the army 1
```

Listing 14: Commit: Subiendo al repositorio la versión final del ejercicio

```
    $ git add .
    $ git commit -m "Eliminando lineas inncesarias"
    $ git push -u origin main
```

4.3. Estructura de laboratorio 01

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

```
lab01/
|--- VideoJuego.java
|--- latex
|--- img
|   |--- logo_abet.png
|   |--- logo_episunsa.png
|   |--- logo_unsa.jpg
|   |--- fragmento_codigoPartel.jpg
|   |--- fragmento_codigoPartell.jpg
|--- programacion_lab01_rescobedoq_v1.0.pdf
|--- programacion_lab01_rescobedoq_v1.0.tex
|--- src
|   |--- 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	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	1	
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://drive.google.com/file/d/1MdWf_3UA_38xs18HmjPk6eH0RzRMJxkX/view?usp=drive_link
- https://drive.google.com/file/d/17xNjwcFykmT8BB5VbZrYEDWUg3UjLlRs/view?usp=drive_link