Informe de Laboratorio 01

Tema: Git y GitHub

Nota

Estudiante	Escuela	Asignatura
Luis Guillermo Luque Condori lluquecon@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Laboratorio de Programación Semestre: II Código: 20233478

Laboratorio	Tema	Duración
01	Git y GitHub	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 18 Agosto 2023	Al 20 Agosto 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)
- 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 de 64 bits, procesador basado en x64.
- Visual Studio Code.
- Latex.
- git version 2.41.0.windows.1
- 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/Choflis/fp2-23b.git`
- URL para el laboratorio 01 en el Repositorio GitHub.
- `https://github.com/Choflis/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

```
1. mkdir lluquecon
```

Listing 2: Dirijéndonos al directorio de trabajo

```
1. cd lluquecon
```

Listing 3: Creando directorio para repositorio GitHub

```
1. mkdir fp2-23b
```

Listing 4: Inicializando directorio para repositorio GitHub

```
1. pwd

2. mkdir lluquecon

3. CONTROL + L

4. cd lluquecon
```

```
5. mkdir fp2-23b

6. cd fp2-23b

7. git init

8. ls -la

9. git config --global user.name "Luis Guillermo Luque Condori"

10. git config --global user.email lluquecon@unsa.edu.pe

11. git config --list

12. mkdir fase01

13. mkdir fase02

14. mkdir fase03

15. cd fase01

16. mkdir lab01

17. cd lab01

18. vim VideoJuego.java

19. cd ..

20. vim .gitignore
```

4.2. Commits

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

Listing 5: Creando .gitignore

```
$ vim lab01/.gitignore
```

Listing 6: lab01/.gitignore

```
*.class
```

Listing 7: Commit: Creando .gitignore para archivos *.class

```
$ git add .
$ git commit -m "Creando .gitignore para archivos *.class"
$ git push -u origin main
```

- En el primer commit creamos gitignore para que evite los archivos tipo class, ademas agregamos un archivo "VideoJuego.java"para verificar la funcionalidad de gitignore.

Listing 8: Ejercicio01.java

```java
import java.util.*;
class Ejercicio01{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Ingrese el nombre de los soldados: ");
        String soldado1 = sc.nextLine();
        String soldado2 = sc.nextLine();
        String soldado3 = sc.nextLine();
        String soldado4 = sc.nextLine();
        String soldado5 = sc.nextLine();

        System.out.println("Nombre del soldado: " + soldado1);
        System.out.println("Nombre del soldado: " + soldado2);
        System.out.println("Nombre del soldado: " + soldado3);
        System.out.println("Nombre del soldado: " + soldado4);
        System.out.println("Nombre del soldado: " + soldado5);

    }

}
```

- La primera actividad nos pide que creemos 5 soldados sólo considerando su nombre, donde sólo podemos usar variables simples.
- Primero importamos la clase Scanner para poder nombrar a cada uno de los cinco soldados.
- Por último imprimimos para que nos muestre el nombre de los soldados.
- Solo se uso el commit solo una vez para poder guardar el avance de la tarea.

Listing 9: Ejecución del Ejercicio01

```
Ingrese el nombre de los soldados:
   Pedro
   Juan
   Luis
   Pancho
   Juarez
   Nombre del soldado: Pedro
   Nombre del soldado: Juan
   Nombre del soldado: Luis
   Nombre del soldado: Pancho
   Nombre del soldado: Juarez
```

Listing 10: Commit: Ejercicio01(Primer commit para poder observar en el git)

```
- git add .
- git commit -m "Ejercicio01(Primer commit para poder observar en el git)"
- git push -u origin main
```

Listing 11: Ejercicio02.java

```java
import java.util.*;
public class Ejercicio02 {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Ingrese el nombre de los soldados: ");
        String soldado1 = sc.nextLine();
        String soldado2 = sc.nextLine();
        String soldado3 = sc.nextLine();
        String soldado4 = sc.nextLine();
        String soldado5 = sc.nextLine();

        System.out.println("Nombre del soldado: " + soldado1 + "\n" + "Vida: 15");
        System.out.println("Nombre del soldado: " + soldado2 + "\n" + "Vida: 20");
        System.out.println("Nombre del soldado: " + soldado3 + "\n" + "vida: 10");
        System.out.println("Nombre del soldado: " + soldado4 + "\n" + "vida: 5");
        System.out.println("Nombre del soldado: " + soldado5 + "\n" + "vida: 13");
    }
}
```

- Para el Ejercicio02 nos pedía crear soldados y como datos tener su nombre y su nivel de vida.
- Primero usamo la clase Scanner para poder nombrar a cada soldado.
- Por último imprimimos pero le ponemos su nivel de vida.
- "Pude realizar que el usuario ponga el nivel de vida, pero puse el nivel de vida predeterminado para cada soldado"
- Solo se uso el commit solo una vez para poder guardar el avance de la tarea.

Listing 12: Ejecución del Ejercicio02

```
Ingrese el nombre de los soldados:
Luis
Pancho
Ronald
Rodrigo
Victor
Nombre del soldado: Luis
Vida: 15
Nombre del soldado: Pancho
Vida: 20
Nombre del soldado: Ronald
vida: 10
Nombre del soldado: Rodrigo
vida: 5
Nombre del soldado: Victor
vida: 13
```

Listing 13: Commit: Ejercicio02(Aun no he usado arreglos)

```
- git add .
- git commit -m "Ejercicio02(Aun no he usado arreglos)"
- git push -u origin main
```

Listing 14: Ejercicio03.java

```java
import java.util.Scanner;

public class Ejercicio03 {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String[] soldados = new String[5];
        for(int i = 0; i < soldados.length; i++){
            System.out.println("ingrese el nombre del soldado " + (i + 1));
            soldados[i] = sc.nextLine();
        }
        String[] imp = new String[5];
        System.out.println("El nombre de los Soldados son: ");
        for(int i = 0; i < soldados.length; i++){
            imp[i] = soldados[i];
            System.out.println("Soldado N" + (i + 1) + ": \n" + imp[i]);
        }
    }
}
```

- En el Ejercicio03 es la versión mejorada del Ejercicio01, ya que ahora usamos arreglos estadar.
- Primero creamos el arreglo de string llamado soldados, esto para pedir el nombre del soldado.
- Después creamos un ciclo for para poder insertar los nombres de cada uno de ellos.
- Por último creamos un ciclo for para imprimir el arreglo.
- Solo se ralizo un commit para poder guardar el avance de ese día, en el commit dije que no podía usar arreglo de clases.

Listing 15: Ejecución del Ejercicio03

```
ingrese el nombre del soldado 1
Luis
ingrese el nombre del soldado 2
Pedro
ingrese el nombre del soldado 3
Roberto
ingrese el nombre del soldado 4
Saul
ingrese el nombre del soldado 5
Juan
El nombre de los Soldados son:
Soldado N1:
Luis
Soldado N2:
Pedro
Soldado N3:
Roberto
Soldado N4:
Saul
Soldado N5:
Juan
```

Listing 16: Commit: Ejercicio03(Empeze a usar arreglos pero no aun clases)

```
  - git add .
  - git commit -m "Ejercicio03(Empeze a usar arreglos pero no aun clases)"
  - git push -u origin main
```

Listing 17: Ejercicio04.java

```java
import java.util.*;
class Ejercicio04{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String[] name = new String[5];
        int[] leave = new int[5];
        System.out.println("Ingrese los datos de los soldados: ");
        name = pedirName(name);
        leave = pedirLeave(leave);
        impr(leave, name);

    }
    public static String[] pedirName(String[] name){
        Scanner sc = new Scanner(System.in);
        for(int i = 0;i < name.length; i++ ){
            System.out.println("Nombre del soldado Numero. " + (i + 1));
            name[i] = sc.nextLine();
        }

        return name;
    }
    public static int[] pedirLeave(int[] leave){
        Scanner sc = new Scanner(System.in);
        for(int i = 0; i < leave.length; i++){
            System.out.println("Ingrese la vida del soldado Nmero. " + (i + 1));
            leave[i] = sc.nextInt();
        }
        return leave;
    }
    public static void impr(int[] leave, String[] name){
        System.out.println("Los datos de los soldados son: ");
        for(int i = 0; i < 5; i++){
            System.out.println("Nombre del soldado Nmero." + (i+1) + ": \n" + name[i]);
            System.out.println("Vida del soldado Nuemro." + (i + 1) + ": \n" + leave[i]);
        }
    }
}
```

- Para el Ejercicio04 use métodos.
- El primer método pide el nombre del soldado.
- El segundo método pide la vida del soldado.
- El tercer método imprime el nombre y la vida del soldado.
- En el commit enviado puse una observación personal sobre el uso de métodos y el avance del día.

Listing 18: Ejecución del Ejercicio04

```
Ingrese los datos de los soldados:
Nombre del soldado Numero. 1
2
Nombre del soldado Numero. 2
Luis
Nombre del soldado Numero. 3
Pedro
Nombre del soldado Numero. 4
Fan
Nombre del soldado Numero. 5
Lu
Ingrese la vida del soldado Nmero. 1
5
Ingrese la vida del soldado Nmero. 2
5
Ingrese la vida del soldado Nmero. 3
5
Ingrese la vida del soldado Nmero. 4
55
Ingrese la vida del soldado Nmero. 5
5
Los datos de los soldados son:
Nombre del soldado Nmero.1:
2
Vida del soldado Nuemro.1:
5
Nombre del soldado Nmero.2:
Luis
Vida del soldado Nuemro.2:
5
Nombre del soldado Nmero.3:
Pedro
Vida del soldado Nuemro.3:
5
Nombre del soldado Nmero.4:
Fan
Vida del soldado Nuemro.4:
55
Nombre del soldado Nmero.5:
Lu
Vida del soldado Nuemro.5:
5
```

Listing 19: Commit: Ejercicio04(Aun se me dificulta usar metodos pero esta vez pude utilizarlos, aun no use arreglo de objetos, cree intento.java para verificar si esta bien la clase intento(aun no entiendo bien como se hace y el porque))

```
- git add .
- git commit -m "Ejercicio04(Aun se me dificulta usar metodos pero esta vez pude
    utilizarlos, aun no use arreglo de objetos, cree intento.java para verificar si esta
    bien la clase intento(aun no entiendo bien como se hace y el porque))"
- git push -u origin main
```

Listing 20: Ejercicio05.java

```java
import java.util.*;
public class Ejercicio05 {
    public static void main (String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Ingrese el nombre del Jugador 1: ");
        String jugador1 = sc.nextLine();
        System.out.println("Ingrese el nombre del Jugador 2 : ");
        String jugador2 = sc.nextLine();
        System.out.println("Ingrese el nmero de soldados del jugador uno: ");
        System.out.println("Solo se permite hasta 5 soldados en la batalla");
        int num1 = sc.nextInt();
        if(num1 > 5){
            System.out.println("Solo se permite numeros del 1 al 5");
        }

        System.out.println("Ingrese el nmero de soldados del jugador dos: ");
        int num2 = sc.nextInt();
        if(num2 > 5){
            System.out.println("Solo se permite numeros del 1 al 5");
        }

        String[] soldados1 = new String[num1];
        String[] soldados2 = new String[num2];
        System.out.println("soldados jugador 1");
        sold1(soldados1);
        System.out.println("Soldados jugador 2");
        sold2(soldados2);
        impGanador(num1, num2, jugador1, jugador2);

    }
    public static void sold1(String[] soldados1){
        for(int i = 0; i < soldados1.length; i++){
            System.out.println("Soldado0" + (i + 1));
        }
    }
    public static void sold2(String[] soldados2){
        for(int i = 0; i < soldados2.length; i++){
            System.out.println("Soldado0" + (i + 1));
        }
    }
    public static void impGanador(int num1, int num2, String jugador1, String jugador2){
        if(num1 > num2){
          System.out.println("gan: " + jugador1);
        }
        if(num1 == num2){
          System.out.println("Empate");
        }
        if(num1 < num2){
          System.out.println("gan: " + jugador2);
        }

    }
}
```

- En este Ejercicio usamos de todo lo aprendido en los otros problemas, usando mejores métodos y haciendo que nos de un ganador.
- Pudo ser mejor, ya que solo gana el que tenga mas soldados.

Listing 21: Ejecución del Ejercicio03

```
Ingrese el nombre del Jugador 1:
Luis
Ingrese el nombre del Jugador 2 :
Lucho
Ingrese el nmero de soldados del jugador uno:
Solo se permite hasta 5 soldados en la batalla
5
Ingrese el nmero de soldados del jugador dos:
4
soldados jugador 1
Soldado01
Soldado02
Soldado03
Soldado04
Soldado05
Soldados jugador 2
Soldado01
Soldado02
Soldado03
Soldado04
 gan : Luis
```

Listing 22: Commit: Aun puede mejorar usando mejores metodos para comprobar mas cosas del Ejercicio05 pero por ahora cumple la funsion de tener un ganador, ademas estoy confiando en el usuario

```
- git add .
- git commit -m "Aun puede mejorar usando mejores metodos para comprobar mas cosas del
    Ejercicio05 pero por ahora cumple la funsion de tener un ganador, ademas estoy
    confiando en el usuario"
- git push -u origin main
```

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	1	
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	1	
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	1	
Total		20		13	

6. Referencias

- https://www.w3schools.com/java/default.asp
- https://www.geeksforgeeks.org/insertion-sort/