Informe de Laboratorio 02
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
02	Arreglos Estandar	04 horas

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

1. Tarea

- En este ejercicio se le solicita a usted implementar el juego del ahorcado utilizando el código parcial que se le entrega.
- Usted debe realizar varios commits y al término de la actividad deberá realizar un informe

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/lab02`

4. Actividades con el repositorio GitHub

4.1. Commits

Listing 1: Primer Commit Creando archivo Ejercicio02.java con el código incompleto

```
cd fase01/lab02
vim Ejercicio02.java
git add Ejercicio02.java
git commit -m "Codigo incompleto del juego de El ahorcado"
git push -u origin main
```

Listing 2: Completando el método ingreseLetra

```
vim Ejercicio02.java
```

- En el código base, la mayor parte del método está completo, sin embargo falta validar si el carácter ingresado es una letra, por lo que en la parte del while, se agrega como condición que mientras la variable laLetra no sea una letra, se seguirá insistiendo al usuario que ingrese dicho valor de manera correcta.
- Para lograr dicha validación, se recurre a los método Character.isLetter(char c) y charAt(int i).

```
public static String ingreseLetra(){
    String laLetra;
    Scanner sc = new Scanner(System.in);
    System.out.println("Ingrese letra: ");
    laLetra = sc.next();
    while(laLetra.length()!= 1 || !Character.isLetter(laLetra.charAt(0))){
        System.out.println("Ingrese letra: ");
        laLetra = sc.next();
    }
    return laLetra;
}
```

Listing 3: Probando el método ingreseLetra

```
javac Ejercicio02.java
java Ejercicio02.java
+---+
|   |
|
|
|
|
=========

- - - - - - - - - - -

Ingrese letra:
2
Ingrese letra:
1
```

Listing 4: Commit: Subiendo al repositorio Ejercicio02.java

```
git add .
git commit -m "Metodo ingreseLetra() completo"
git push -u origin main
```

Listing 5: Completando el método letraEnPalabraSecreta

```
vim Ejercicio02.java
```

- En el código base, este método está vacío, por lo que se debe completar.
- Este método básicamente recibe dos parámetros, un String que es el valor que el usuario ingresa y el otro sería la palabra que se extrajo del arreglo de palabras iniciales, para implementar el código, se recurre a un ciclo for el cual recorre la palabra elegida extrayendo los carácteres de dicha palabra y comparándo cada una con el caracter ingresado por el usuario. Para comparar la letra (dato de tipo String) y el carácter extraído de la palabra (dato de tipo char) se opta por convertir el carácter en String usando el método Character.toStrng(char c) para luego usar equasl() el cual compara String. Si hay coincidencias se retorna true, de lo contrario se retorna false

```
public static boolean letraEnPalabraSecreta(String letra, String palSecreta ){
    for(int i = 0; i < palSecreta.length();i++){
        if(Character.toString(palSecreta.charAt(i)).equalsIgnoreCase(letra))
            return true;
    }
    return false;
}
```

Listing 6: Commit: Subiendo al repositorio Ejercicio02.java

```
git add .
git commit -m "Metodo letraEnPalabraSecreta realizado"
git push -u origin main
```

Listing 7: Creando un nuevo método que genera un arreglo de Strings formado por Subguiones

```
vim Ejercicio02.java
```

- En el código base, este método no está implementado.
- Este método se encarga de generar un arreglo de Subguiones, el tamaño es de acuerdo a la palabra elegida, el propósito de este método es que el arreglo que se forme formará parte del parámetro del método mostrarBlancosActualizados

```
124     public static String[] arregDeSubguiones(String str){
125         String[] arraySubguiones = new String[str.length()];
126         for(int i = 0; i <str.length();i++){
127             arraySubguiones[i] = "_ ";
128         }
129         return arraySubguiones
130     }
```

Listing 8: Commit: Creando un método que genera un arreglo de subguiones de con tamaño de acuerdo a la palabra elegida

```
git add Ejercicio02.java
git commit -m "Creando un metodo que genera un arreglo de subguiones de con tamano de
    acuerdo a la palabra elegida"
```

Listing 9: Corrigiendo los parámatros del método mostrarBlancosActualizados

```
vim Ejercicio02.java
```

- En el código base, este método recibe como único parámetro un String el cual es la letra que el usuario ingresa.
- El método así como está, no aporta mucho a la solución del problema, es por ello que decido agregarle dos parámetros más, los cuales son un arreglo de String (este arreglo es el de los Subguiones) y adicionalmente se le agrega un String (se trata de la palabra elegida)

```
119     public static void mostrarBlancosActualizados(String palabrajuego, String letra, String[] array){
120         System.out.println("PROCESANDO.....");
121         for(int i =0; i< palabraJuego.length();i++){
122             if(Character.toString(palabraJuego.charAt(i)).equals(letra) ){
123                 arregloVacio[i] = letra;
124             }
125         }
126         for( int i = 0; i < arregloVacio.length; i++){
127             System.out.print(arregloVacio[i]+" " );
128         }
129     }
```

Listing 10: Commit: Aumentando parámetros y completando el método de mostrarBlancosActualizados

```
git add Ejercicio02.java
git commit -m "Aumentando parametros y completando el metodo de
    mostrarBlancosActualizados"
git push -u origin main
```

Listing 11: Aumentando líneas del main y determinando si se ganó o no

```
vim Ejercicio02.java
```

- En la línea 74 se recibe un arreglo generado por un método. Este arreglo contiene estos elementos
- En el ciclo while, se va actualizando la imagen de acuerdo a lo que el usuario ingrese. En caso el usuario logre adivinar todas las letras en los turnos previstos, el ciclo se acabará y otra forma en la que se sale ciclo es cuando se cumplen con los turnos ya establecidos

```
74          String[] arregloGuiones = arregloVacio(palSecreta);
75
76          System.out.println("\n");
77          while(contador <= 6){
78              letra = ingreseLetra();
79              if (letraEnPalabraSecreta(letra, palSecreta)){
80                  mostrarBlancosActualizados(palSecreta, letra, arregloGuiones );
81                  if(!String.join("", arregloGuiones).contains("_")) {
82                      System.out.println("FELICIDADES, GANÓ EN "+ contador+ " turnos");
83                      break;
84                  }
85              }else{
86                  System.out.println(figuras[contador]);
87                  contador = contador +1;
88                  mostrarBlancosActualizados(palSecreta, letra, arregloGuiones);
89              }
90              if(contador == 7){
91                  System.out.println("PERDIÓ");
92              }
93
94          }
```

Listing 12: Compilando Ejercicio02.java en su versión final

```
javac Ejercicio02.java
java Ejercicio02
+---+
|   |
|
|
|
|
=========
- - - - - - -

Ingrese letra:
o
+---+
|   |
O   |
|
|
|
=========
PROCESANDO.....
-  -  -  -  -  -  -
Ingrese letra:
c
+---+
|   |
O   |
|   |
|
|
=========
PROCESANDO.....
-  -  -  -  -  -  -
Ingrese letra:
p
PROCESANDO.....
p  -  -  -  -  -  -
Ingrese letra:
```

```
r
PROCESANDO.....
p r _ _ _ _ _
Ingrese letra:
u
PROCESANDO.....
p r u _ _ _ _
Ingrese letra:
e
PROCESANDO.....
p r u e _ _ _
Ingrese letra:
b
PROCESANDO.....
p r u e b _ _
Ingrese letra:
a
PROCESANDO.....
p r u e b a _
Ingrese letra:
s
PROCESANDO.....
p r u e b a s
FELICIDADES, HA GANADO EN 3 turnos
```

Listing 13: Commit: Determinando el ganador y haciendo mejoras en el código

```
$ git add .
$ git commit -m "Determinando el ganador y haciendo mejoras en el codigo"
$ git push -u origin main
```

4.2. Estructura de laboratorio 01

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

```
lab02/
|--- Ejercicio02.java
|--- latex
|----- img
|     |--- logo_abet.png
|     |--- logo_episunsa.png
|     |--- logo_unsa.jpg
|     |--- arregloSubguiones.jpg
|     |--- ganador.jpg
|     |--- letraEnPalabraSecreta.jpg
|     |--- metodoIngreseLetra.jpg
|----- src
|--- programacion_lab02_rescobedoq_v1.0.pdf
|--- programacion_lab02_rescobedoq_v1.0.tex
```

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