Informe de Laboratorio 02
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
02	Arreglos Estandar	04 horas

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

1. Tarea

- En este ejercicio se le solicita a usted implementar el juego del ahorcado utilizando el código parcial que se le entrega. Deberá considerar que:
  - El juego valida el ingreso de letras solamente.
  - En caso el usuario ingrese un carácter equivocado le dará el mensaje de error y volverá a solicitar el ingreso
  - El juego supone que el usuario no ingresa una letra ingresada previamente
  - El método ingreseLetra() debe ser modificado para incluir las consideraciones de validación
  - Puede crear métodos adicionales

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 02 en el Repositorio GitHub.
- `https://github.com/JhonatanDczel/fp2-23b/tree/main/fase01/lab02`

4. Commit Principal

Listing 1: Commit principal, sacado de git log

```
commit ac4b27bed13d9852c0bd8796cf1562711052890d (HEAD -> main, origin/main)
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Wed Sep 20 16:47:43 2023 -0500

    Laboratorio 2: El juego del ahorcado
```

5. Metodo getPalabraSecreta

5.1. Creacion del metodo

- Necesitamos un metodo que tenga como funcion la de escoger un numero al azar y generar con ello un palabra al azar como palabra secreta
- Para eso debera recibir un arreglo de Strings como entrada y devolver un String unico como salida

Listing 2: Metodo getPalabraSecreta

```
public static String getPalabraSecreta(String[] lasPalabras){
    int ind;
    int indiceMayor = lasPalabras.length - 1;
    int indiceMenor = 0;
    ind = (int) (Math.random() * (indiceMayor - indiceMenor + 1)) + indiceMenor;
    return lasPalabras[ind];
}
```

6. Metodo mostrarPalabra

- Dibujar el contenido de un arreglo que contiene nuestra palabra
- Lo haremos con un simple bucle for each, para recorrer el arreglo mientras imprimimos el valor
- Adicionalmente los separaremos por espacios para darle voluimen a nuestra palabra

Listing 3: Creando un nuevo atributo para Soldado

```
public static void mostrarPalabra(char[] palabra){
    for(char c : palabra){
        System.out.print(c + " ");
    }
    System.out.println();
}
```

6.1. Ejecucion

Listing 4: Ejecucion del codigo

```
a
_ _ _ _ _ a _ a _ _ _ _
```

- Asi es como se ve la ejecucion en consola, con un ejemplo generico

7. Metodo ingreseLetra

7.1. Las consideraciones

- Necesitamos verificar la entrada del usuario para asegurarnos de que sea un solo caracter y que sea una letra
- Para eso, nos apoyamos de un metodo auxiliar, para verificar si la entrada es una letra:

Listing 5: metodo checker()

```
public static boolean checker(String letra){
    String[] abc = {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
        "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"};
    for(String letraAbc : abc){
        if(letraAbc.equals(letra)){
            return false;
        }
    }
    return true;
}
```

7.2. La condicion de salida

Listing 6: Metodo ingreseLetra

```
public static String ingreseLetra(){
    String laLetra;
    Scanner sc = new Scanner(System.in);
    System.out.println("Ingrese letra: ");
    laLetra = sc.next().toLowerCase();
    while (laLetra.length() != 1 || checker(laLetra)){
        System.out.println("Ingrese una sola letra: ");
        laLetra = sc.next().toLowerCase();
```

```
        }
        return laLetra;
    }
```

- Como podemos ver, el metodo se asegura de permitir unicamente el paso de letras
- Para trabajar sin distinciones entre mayusculas y minusculas usamos el metodo toLowerCase, para trabajar estandarizadamente y evitar conflictos

8. Metodo letraEnPalabraSecreta

- Ahora necesitamos saber si la letra que ingreso el usuario esta contenida en la palabra secreta

Listing 7: Implementacion del metodo letraEnPalabraSecreta

```
    public static boolean letraEnPalabraSecreta(String letra, String palSecreta, char[]
        palabraMostrada){
        boolean letraAdivinada = false;
        for(int i = 0; i < palSecreta.length(); i++){
            if(palSecreta.charAt(i) == letra.charAt(0)){
                if(palabraMostrada[i] == '_'){
                    palabraMostrada[i] = letra.charAt(0);
                    letraAdivinada = true;
                }
            }
        }
        return letraAdivinada;
    }
```

- Recorremos la palabra secreta en busca de la letra que el usuario ingreso, de ser ese el caso, se retorna la letra adivinada.

9. Metodo contarLetrasRestantes

- A este punto necesitamos un metodo que se encargue de verificar cuantas letras quedan por descubrir, para poder finalizat el programa en caso de llegar a 0

Listing 8: Metodo contarLetrasRestantes

```
    public static int contarLetrasRestantes(char[] palabraMostrada){
        int contador = 0;
        for(char c : palabraMostrada){
            if (c == '_'){
                contador++;
            }
        }
        return contador;
    }
```

- Recorremos el arreglo en busca de caracteres '-', en caso de encontrarlo, el contador sube
- Al final, el metodo devuelve el numero de letras que quedan por adivinar

10. Construccion del metodo main

- Ahora necesitamos armar todos los componentes para crear el juego, empezamos inicializando variables

Listing 9: Metodo display()

```
int contador = 0;
String letra;
String[] palabras = { "programacion", "java", "indentacion", "clases", "objetos",
    "desarrollador", "pruebas", "test", "pruebita"};
String palSecreta = getPalabraSecreta(palabras);
int letrasRestantes = palSecreta.length();

System.out.println(figuras[contador]);
char[] palabraMostrada = new char[palSecreta.length()];
for (int i = 0; i < palabraMostrada.length; i++){
    palabraMostrada[i] = '_';
}
mostrarPalabra(palabraMostrada);
```

- La variable contador, sera quien nos indique cuantas veces el usuario ha ingresado una respuesta erronea
- Entre otras inicializaciones, tenemos la de palabraMostrada, que rellenamos de caracteres '-' haciendo uso de un ciclo for

10.1. Estructura del juego

- Todo el juego estara contenido en un ciclo while, cuya condicion de salida sera que el contador sea menor que 6 y que queden mas de 0 letras por adivinar

Listing 10: Metodo principal

```
while(contador < 6 && letrasRestantes > 0){
    letra = ingreseLetra();
    char letraChar = letra.charAt(0);
    if(letrasAdivinadas[letraChar - 'a']){
        System.out.println("Ya has ingresado la letra " + letra);
    }else{
        letrasAdivinadas[letraChar - 'a'] = true;
        if(letraEnPalabraSecreta(letra, palSecreta, palabraMostrada)){
            mostrarPalabra(palabraMostrada);
            letrasRestantes = contarLetrasRestantes(palabraMostrada);
        }else{
            contador++;
            System.out.println(figuras[contador]);
        }
    }
}
```

- El juego se mantendra en marcha mientras no se hayan alcanzado el llimite de jugadas erroneas, o no se hayan terminado las letras pr adivinar, es decir, adivino la palabra

11. Ejecucion

- Tendremos ahora una demostracion sobre el modo de juego del Ahorcado:

Listing 11: Ejecucion en la linea de comandos

```
 +---+
 |   |
     |
     |
     |
     |
=========
_ _ _ _
Ingrese letra:
e
_ e _ _
Ingrese letra:
t
t e _ t
Ingrese letra:
s
t e s t

 Ganaste ! La palabra secreta es: test
```

- Como hemos podido ver, hemos acertado la palabra y parece estar correcto el sistema de deteccion de la palabra
- Ahora haremos un test sobre las limitaciones con la entrada de datos

Listing 12: Ejecucion por terminal del juego del Ahorcado

```
 +---+
 |   |
     |
     |
     |
     |
=========
_ _ _ _ _ _ _
Ingrese letra:
3
Ingrese una sola letra:
3
Ingrese una sola letra:
f
 +---+
 |   |
 O   |
     |
     |
     |
=========
Ingrese letra:
```

```
f
Ya has ingresado la letra f
Ingrese letra:
```

- Como vemos, la deteccion de entradas incorrectas esta funcionando bien
- Se probo con numeros como entrada, y con letras repetidas
- Si no se ingresan entradas legales, el programa seguira indefinidamente hasta que se ingrese una jugada legal

12. Rúbricas

12.1. 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.

12.2. 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	
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	