Informe de Laboratorio 02
Tema: Arreglos estándar

Nota

Estudiante	Escuela	Asignatura
Ryan Fabian Valdivia Segovia rvaldiviase@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la programación 2 Semestre: II Código: 1701213

Laboratorio	Tema	Duración
02	Arreglos estándar	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 11 Septiembre 2023	Al 17 Septiembre 2023

1. Tarea

1.1. Actividad: Juego del Ahorcado

- 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 Windows 11 Home Single Language 64 bits 22621.2283
- VIM 9.0.
- Visual Studio Code 64 bits 1.82.2
- OpenJDK 64-Bits 11.0.16.1
- Git 2.41.0.windows.1
- Cuenta en GitHub con el correo institucional.
- Código parcial proporcionado por el profesor.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/RyanValdivia/fp2-23b.git`
- URL para el laboratorio 01 en el Repositorio GitHub.
- `https://github.com/RyanValdivia/fp2-23b/tree/main/fase01/lab02`

4. Actividades

- Realicé un commit copiando el código parcial que nos proporcionó el profesor.

Listing 1: Comentando el código paricial

```
$ git log Ahorcado.java
commit 55760b4d4357c7d051424f287a351230595960c8
Author: RYAN VALDIVIA <rvaldiviase@unsa.edu.pe>
Date:  Tue Sep 19 11:06:57 2023 -0500
   Actividad del ahorcado: Copiando el codigo parcial proporcionado
```

Listing 2: Código parcial

```
6   import java.util.*;
7
8  public class Ahorcado {
9      public static void main(String[] args) {
10
11         String ahor1 = " +---+ \n" +
12                 " |   | \n" +
13                 "     | \n" +
14                 "     | \n" +
15                 "     | \n" +
16                 "     | \n" +
17                 "========= ";
18         String ahor2 = " +---+ \n" +
19                 " |   | \n" +
20                 " O   | \n" +
21                 "     | \n" +
22                 "     | \n" +
23                 "     | \n" +
24                 "========= ";
25         String ahor3 = " +---+ \n" +
26                 " |   | \n" +
27                 " O   | \n" +
28                 " |   | \n" +
29                 "     | \n" +
30                 "     | \n" +
31                 "========= ";
32         String ahor4 = " +---+ \n" +
33                 " |   | \n" +
34                 " O   | \n" +
35                 "/|   | \n" +
36                 "     | \n" +
```

```
37              "     | \n" +
38              "========= ";
39          String ahor5 = " +---+ \n" +
40                  " |   | \n" +
41                  " O   | \n" +
42                  "/|\\  | \n" +
43                  "     | \n" +
44                  "     | \n" +
45                  "========= ";
46          String ahor6 = " +---+ \n" +
47                  " |   | \n" +
48                  " O   | \n" +
49                  "/|\\  | \n" +
50                  "/    | \n" +
51                  "     | \n" +
52                  "========= ";
53          String ahor7 = " +---+ \n" +
54                  " |   | \n" +
55                  " O   | \n" +
56                  "/|\\  | \n" +
57                  "/ \\  | \n" +
58                  "     | \n" +
59                  "========= ";
60
61          String[] figuras = { ahor1, ahor2, ahor3, ahor4, ahor5, ahor6, ahor7 };
62          int contador = 1;
63          String letra;
64          String[] palabras = { "programacion", "java", "identacion", "clases", "objetos",
                "desarrollador", "pruebas" };
65          String palSecreta = getPalabraSecreta(palabras);
66          System.out.println(figuras[0]);
67          mostrarBlancos(palSecreta);
68          System.out.println("\n");
69          while (contador <= 6) {
70              letra = ingreseLetra();
71              if (letraEnPalabraSecreta(letra, palSecreta)) {
72                  mostrarBlancosActualizados(letra);
73              } else {
74                  System.out.println(figuras[contador]);
75              }
76              contador = contador + 1;
77          }
78          // COMPLETAR PARA INDICAR SI GANO, PERDIO Y CUANTOS TURNOS NECESITO
79          System.out.println("\n");
80      }
81
82      public static String getPalabraSecreta(String[] lasPalabras) {
83          String palSecreta;
84          int ind;
85          int indiceMayor = lasPalabras.length - 1;
86          int indiceMenor = 0;
87          ind = (int) ((Math.random() * (indiceMayor - indiceMenor + 1) + indiceMenor));
88          return lasPalabras[ind];
89      }
90
91      public static void mostrarBlancos(String palabra) {
```

```
92          for (int i = 0; i < palabra.length(); i++)
93              System.out.print("_ ");
94
95      }
96
97      public static String ingreseLetra() {
98          String laLetra;
99          Scanner sc = new Scanner(System.in);
100         System.out.println("Ingrese letra: ");
101         laLetra = sc.next();
102         while (laLetra.length() != 1) {
103             System.out.println("Ingrese letra: "); // COMPLETAR PARA VALIDAR CARACTERES
                    PERMITIDOS
104             laLetra = sc.next();
105         }
106         return laLetra;
107     }
108
109     public static boolean letraEnPalabraSecreta(String letra, String palSecreta) {
110         // COMPLETAR
111         return false;
112     }
113
114     public static void mostrarBlancosActualizados(String letra) {
115         // COMPLETAR
116         System.out.println("PROCESANDO.....");
117     }
118 }
```

- Ahora, solo faltaría implementar todo para que el juego sea perfectamente jugable, para ello, comencé por asegurarme de que todos los valores que entren sean un solo caracter y sean letras. Por eso, creé un método para determinar si un string es una letra.

Listing 3: Asegurándome de que solo entren letras

```
119     public static boolean esCaracter(String str) {
120         if (str == null || str.equals("")) {
121             return false;
122         }
123         char c = str.charAt(0);
124         return ('a' <= c && c <= 'z');
125     }
```

- Y solo queda añadirlo como condición al método para ingresar letras.

Listing 4: Método ya armado y preparado

```
126     public static String ingreseLetra() {
127         String laLetra;
128         Scanner sc = new Scanner(System.in);
129         System.out.println("Ingrese letra: ");
130         laLetra = sc.next();
131         while (laLetra.length() != 1 || !esCaracter(laLetra)) {
132             System.out.println("Ingrese solo letras, vuelva a intentarlo");
```

```
            laLetra = sc.next();
        }
        return laLetra;
    }
```

- Asímismo, implementé el método para saber si la letra ingresada está en la palabra secreta.

Listing 5: Letra en palabra secreta

```
    public static boolean letraEnPalabraSecreta(String letra, String palSecreta) {
        return palSecreta.indexOf(letra) != -1;
    }
```

- Ahora viene lo complicado, establecer la lógica interna del juego del ahorcado.
- Para esto, pensé en tener tres cosas con las qué trabajar. En primer lugar, creé un método que cree un tipo de "blacklist.º lista negra, para llevar la cuenta de las letras que le falta ingresar al usuario para poder ganar.

Listing 6: Creando una blacklist

```
    public static String getBlacklist(String str) {
        String blacklist = "" + str.charAt(0);
        for (int i = 0; i < str.length(); i++) {
            if (blacklist.indexOf(str.charAt(i)) == -1) {
                blacklist += str.charAt(i);
            }
        }
        return blacklist;
    }
```

- Entonces, a este método se le da la palabra secreta, y devuelve un nuevo string con todas las letras diferentes que tiene que adivinar el usuario.
- Luego, necesitaba una forma de que el usuario supiera cuántas letras tiene la palabra secreta y que, si bien al principio salga vacía, se vaya actualizando conforme el usuario ingrese las letras correspondientes.
- Para esto, pensé en trabajar esto como arreglos de caracteres, creando uno, lleno de guiones bajos para mostrar la palabra vacía, y que se vaya actualizando.

Listing 7: Creando la plantilla de la palabra secreta

```
    public static char[] crearVacio(String str) {
        char[] incognito = new char[str.length()];
        for (int i = 0; i < str.length(); i++) {
            incognito[i] = '_';
        }
        return incognito;
    }
```

- Entonces, el plan es usar este arreglo "vacío.[ei]rlo actualizando, para lograr ese resultado, elaboré otro método para modificar este arreglo e irle añadiendo las letras conforme se vayan ingresando.

Listing 8: Añadiendo las letras a la palabra vacía

```
156 public static char[] modificarArreglo(char[] incognita, char[] palabra, String letra) {
157     char actual = letra.charAt(0);
158     for (int i = 0; i < incognita.length; i++) {
159         if (palabra[i] == actual) {
160             incognita[i] = actual;
161         }
162     }
163     return incognita;
164  }
```

- Este método recibe tres cosas como dominio, el arreglo vacío que ya debió ser creado con antelación, un arreglo de chars que corresponde a la palabra secreta y la letra ingresada por el usuario; entonces, modificará el arreglo vacío para añadir la letra ingresada en los lugares correspondientes en relación a la palabra secreta.
- Una vez que ya están los métodos realizados, es hora de armarlo todo en el método main para construir el juego del ahorcado.

Listing 9: Declarando todo lo necesario

```
165  char[] secreta = palSecreta.toCharArray();
166   char[] incognita = crearVacio(palSecreta);
167   String blacklist = getBlacklist(palSecreta);
168   int turnos = 1;
```

- Inicié declarando las variables requeridas, como la variable ïncognita"que esta llena de guiones bajos, que representarán la palabra vacía, así como la variable "turnos"que llevará la cuenta de cuántos turnos han transcurrido hasta que el usuario ganó.

Listing 10: Código principal

```
169  while (contador <= 6) {
170         mostrarPalabra(incognita);
171         System.out.println();
172         letra = ingreseLetra();
173         if (letraEnPalabraSecreta(letra, blacklist)) {
174             blacklist = quitarLetra(blacklist, letra);
175             incognita = modificarArreglo(incognita, secreta, letra);
176             if (blacklist.length() == 0) {
177                 mostrarPalabra(incognita);
178                 break;
179             }
180         } else {
181             System.out.println(figuras[contador]);
182             contador = contador + 1;
183         }
184         turnos++;
185  }
```

- En este ciclo while se encuentra el código principal del ahorcado, para comenzar, utilicé un método "mostrarPalabra.[el] cual imprimirá el contenido de un arreglo como si fuera una palabra.

Listing 11: Mostrar palabra

```
186 public static void mostrarPalabra(char[] letras) {
187     System.out.println("PROCESANDO.....");
188     for (int i = 0; i < letras.length; i++) {
189         System.out.print(letras[i] + " ");
190     }
191     System.out.println();
192 }
```

- Esto permitirá que el usuario lleve una cuenta de las letras que le falta adivinar, así como la longitud de la palabra, al mostrar la palabra vacía al comenzar el juego.
- Después de eso, leo la letra a ingresar por el usuario por primera vez y utilizo un condicional con dos posibles resultados.
- Si la letra está en la palabra secreta, entonces pasa a modificar la blacklist con el método "quitarLetra".

Listing 12: Método quitar letra

```
193 public static String quitarLetra(String blacklist, String letra) {
194     int idx = blacklist.indexOf(letra);
195     blacklist = blacklist.substring(0, idx) + blacklist.substring(idx + 1);
196     return blacklist;
197 }
```

- Esto con la finalidad de modificar la blacklist y remover la letra que ya ha salido. Pudiendo así llevar un control de las letras que faltan por salir.
- Después de ello, se modifica el arreglo incognita, para poder reemplazar los guiones bajos con la letra que ingresó, para poder ir completando la palabra, usando el método "modificar Arreglo", para que al dar otra vuelta al ciclo, se imprima el arreglo modificado y el jugador pueda saber cuantos espacios de letras le falta por ingresar.
- Además, se coloca un último condicional para establecer la condición de victoria, si la blacklist se queda vacía, eso significa que ya no quedan letras por ingresar, entonces el juego se termina, para ello, usamos un break.
- Todos estos cambios fueron realizándose en un tiempo determinado, pero los avances más importantes son en el penúltimo commit de github.

Listing 13: Subiendo muchos cambios y creaciones de métodos

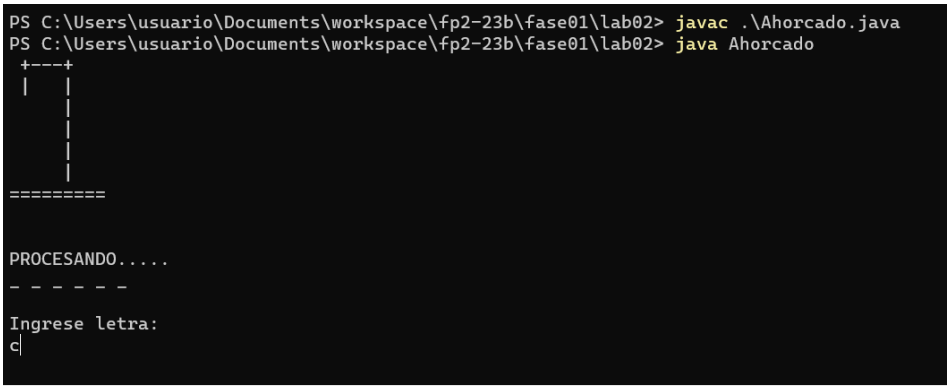
```
$ git log Ahorcado.java
commit df63390b5491ee4af2d993bfdb6dfe896f0d2e1f
Author: RYAN VALDIVIA <rvaldiviase@unsa.edu.pe>
Date:  Tue Sep 19 22:12:36 2023 -0500
   Version de prueba, se crearon diferentes metodos para realizar comprobaciones y que
       sea un juego del ahorcado completamente jugable y sea visual para el usuario
```

- Para posteriormente ser subida la versión final en el último commit.

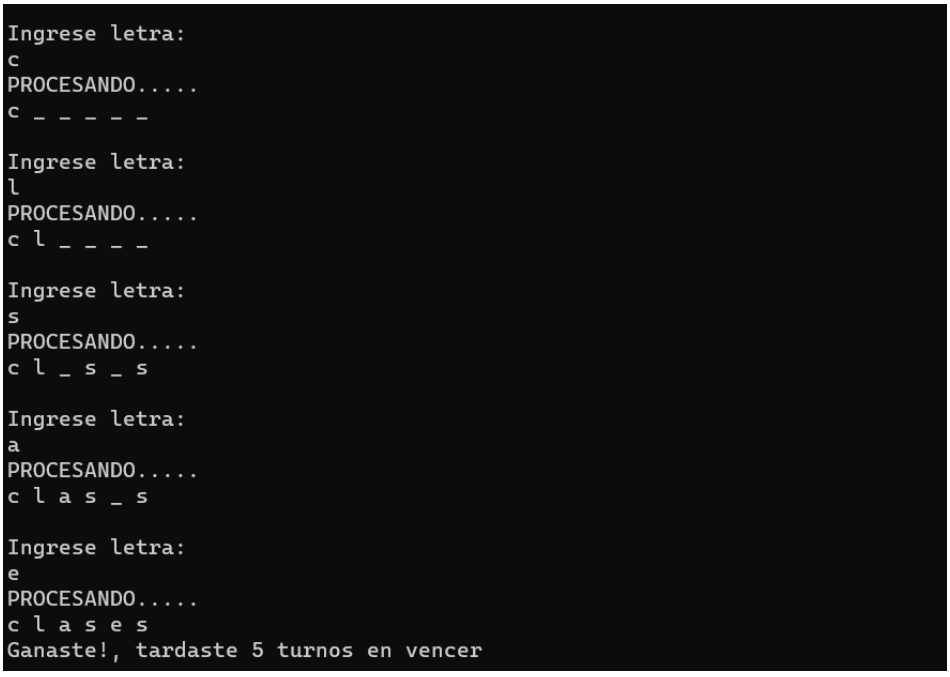
Listing 14: La versión final

```
$ git log Ahorcado.java
commit 7afcc7d1b4a8bb7fb194c03737868eea4be5ecbf
Author: RYAN VALDIVIA <rvaldiviase@unsa.edu.pe>
Date:  Tue Sep 19 22:24:48 2023 -0500
   Version final del codigo del ahorcado, quitando las depuraciones y haciendo algunos
       ajustes mas
```

- A continuación, algunas capturas de pantalla de la ejecución del código.



- Al ingresar una letra y acertar hasta terminar la palabra, pasa lo de la siguiente captura.



- Y al perder, pasa lo siguiente.

```
Ingrese letra:
s
 +---+
 |   |
 O   |
/|\  |
/    |
     |
=========
PROCESANDO.....
_ _ o _ _ _ m _ c _ o _

Ingrese letra:
d
 +---+
 |   |
 O   |
/|\  |
/ \  |
     |
=========
Perdiste, prueba de nuevo
```

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	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	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	4	
Total		20		18	

6. Referencias