
	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 1</p>

## INFORME DE LABORATORIO

### (formato estudiante)



INFORMACIÓN BÁSICA					
<b>ASIGNATURA:</b>	<i>Laboratorio Fundamentos de la programación 2</i>				
<b>TÍTULO DE LA PRÁCTICA:</b>	<i>Arreglos estándar</i>				
<b>NÚMERO DE PRÁCTICA:</b>	<i>2</i>	<b>AÑO LECTIVO:</b>	<i>1</i>	<b>NRO. SEMESTRE:</b>	<i>//</i>
<b>FECHA DE PRESENTACIÓN</b>	<i>27/09/2024</i>	<b>HORA DE PRESENTACIÓN</b>	<i>16/40/00 pm</i>		
<b>INTEGRANTE (s)</b> <i>Hilacondo Begazo, Emanuel David</i>				<b>NOTA (0-20)</b>	
<b>DOCENTE(s):</b> <i>Ing. Lino José Pinto Oppe</i>					

RESULTADOS Y PRUEBAS
<p><b>I. EJERCICIOS RESUELTOS:</b></p> <p>Primeramente, en el <i>main</i> se desarrollan las imágenes que se mostrarán a lo largo del juego. En este caso, al ser un juego de ahorcados, cada vez que no se ingrese una letra que pertenezca a la palabra, se mostrará la evolución.</p> <pre> 1 package Laboratorio2; 2 /*Laboratorio Nr2 - Ejercicio1 3  *Autor: Hilacondo Begazo,Emanuel David 4  *Colaboró: con nadie 5  *Tiempo: -- 6  */ 7 import java.util.Scanner; 8 9 public class ejercicio1 { 10     public static void main(String[] args) { 11         // Representaciones gráficas del ahorcado para cada etapa del juego. 12         String ahor1 = " +---+ \n" + 13             "       \n" + 14             "       \n" + 15             "       \n" + 16             "       \n" + 17             "       \n" + 18             "===== "; 19         String ahor2 = " +---+ \n" + 20             "       \n" + 21             " 0     \n" + 22             "       \n" + 23             "       \n" + 24             "       \n" + 25             "===== "; 26         String ahor3 = " +---+ \n" + 27             "       \n" + 28             " 0     \n" + 29             "       \n" + 30             "       \n" + 31             "       \n" </pre>

```

31         "      | \n" +
32         "=====";
33     String ahor4 = " +---+ \n" +
34         " | | \n" +
35         " 0 | \n" +
36         "/| | \n" +
37         " | \n" +
38         " | \n" +
39         "=====";
40     String ahor5 = " +---+ \n" +
41         " | | \n" +
42         " 0 | \n" +
43         "/|\ | \n" +
44         " | \n" +
45         " | \n" +
46         "=====";
47     String ahor6 = " +---+ \n" +
48         " | | \n" +
49         " 0 | \n" +
50         "/|\ | \n" +
51         "/ | \n" +
52         " | \n" +
53         "=====";
54     String ahor7 = " +---+ \n" +
55         " | | \n" +
56         " 0 | \n" +
57         "/|\ | \n" +
58         "/ \ | \n" +
59         " | \n" +
60         "=====";

```

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 3</p>

Aún en el *main*, guardamos todas las etapas del ahorcado y creamos un contador de fallos, además de la variable letra, que será la que seleccione el usuario. También se crea un arreglo con las palabras que estarán en juego.

```

62      // Arreglo que contiene todas las etapas del ahorcado.
63      String[] figuras = {ahor1, ahor2, ahor3, ahor4, ahor5, ahor6, ahor7};
64      int contador = 1; // Contador para fallos.
65      char letra; // Variable para almacenar la letra ingresada por el usuario.
66      String[] palabras = {"programacion", "java", "indentacion", "clases",
67                           "objetos", "desarrollador", "pruebas"};
68

```

En esta parte del *main* creamos dos métodos importantes. El primero elegirá una de las palabras del arreglo anterior, que se usará para el juego. Después, se crea un arreglo que contendrá las letras ingresadas por el usuario que coincidan con las letras de la palabra elegida. El segundo método mostrará los espacios correspondientes a la palabra.

```

69      // Se elige una palabra secreta de forma aleatoria.
70      String palSecreta = getPalabraSecreta(palabras);
71
72      // Se muestra la primera figura del ahorcado y las líneas en blanco.
73      System.out.println(figuras[0]);
74      char[] palabCorrecta = new char[palSecreta.length()];
75
76      // Inicializa y muestra los espacios en blanco correspondientes a la palabra secreta.
77      mostrarBlancos(palabCorrecta);
78      System.out.println("\n");
79

```



Como se mencionó anteriormente, aquí están los métodos. En el primero, se utiliza un algoritmo en el que, con ayuda de `Math.random()`, se genera un número dentro del rango del arreglo de palabras, y este será la palabra jugar.

En el segundo método, se imprimirán los espacios representados por '\_', equivalentes a la cantidad de letras que tiene la palabra.

```

104     // Selecciona una palabra aleatoria del arreglo.
105     public static String getPalabraSecreta(String[] lasPalabras) {
106         int indiceMayor = lasPalabras.length - 1;
107         int indiceMenor = 0;
108         int ind = (int) (Math.random() * (indiceMayor - indiceMenor + 1) + indiceMenor);
109         return lasPalabras[ind];
110     }
111
112     // Muestra los espacios en blanco para la palabra secreta.
113     public static void mostrarBlancos(char[] palabCorrecta) {
114         for (int i = 0; i < palabCorrecta.length; i++) {
115             palabCorrecta[i] = '_';
116             System.out.print(palabCorrecta[i] + " ");
117         }
118     }

```

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 4</p>

Como indica el comentario, el `while` da inicio al juego, el cual funcionará hasta que el contador sea menor que 7. Esto es porque, al llegar a ese número, has perdido. Primero, en este bucle, se llama a un método encargado de recibir una letra del usuario. Luego, con un condicional de tipo booleano, se verifica si la letra ingresada pertenece a la palabra en juego. En caso de ser verdad, se ingresa a un método que actualiza los '\_' por la letra correcta. Si es falso, el contador aumenta y la imagen del ahorcado se actualiza.

```

80      // COMIENZA EL JUEGO
81      while (contador <= 6) {
82          letra = ingreseLetra(); // Solicita al usuario una letra.
83
84          // Si la letra está en la palabra secreta, se actualizan los espacios en blanco.
85          if (letraEnPalabraSecreta(letra, palSecreta))
86              mostrarBlancosActualizados(letra, palSecreta, palabCorrecta);
87          else {
88              // Si la letra no está en la palabra secreta, se muestra la siguiente etapa del ahorcado.
89              System.out.println(figuras[contador]);
90              contador++; // Aumenta el contador de fallos.
91          }
92      }



```

Como se explicó, el primer método recibe la letra del usuario y cuenta con un bucle que se asegura de que lo ingresado sea efectivamente una letra. El segundo método se encarga de verificar si la letra ingresada pertenece a la palabra mediante un condicional, y el tercero actualiza la lista de '\_' reemplazando el espacio correspondiente con la letra ingresada.

```

120      // Solicita al usuario ingresar una letra y la valida.
121      public static char ingreseLetra() {
122          char laLetra = 'n';
123          boolean confirmacion = true;
124          Scanner sc = new Scanner(System.in);
125          while (confirmacion) {
126              System.out.print("\nIngresa letra: ");
127              laLetra = sc.next().toLowerCase().charAt(0);
128              for (char i = 'a'; i <= 'z'; i++) {
129                  if (laLetra == i)
130                      confirmacion = false;
131              }
132          }
133          return laLetra;
134      }
135
136      // Verifica si una letra está en la palabra secreta.
137      public static boolean letraEnPalabraSecreta(char letra, String palSecreta) {
138          for (int i = 0; i < palSecreta.length(); i++) {
139              if (letra == palSecreta.charAt(i))
140                  return true;
141          }
142          return false;
143      }
144
145      // Actualiza y muestra el estado actual de la palabra secreta con las letras acertadas.
146      public static void mostrarBlancosActualizados(char letra, String palabra, char[] palabCorrecta) {
147          for (int j = 0; j < palabCorrecta.length; j++) {
148              if (letra == palabra.charAt(j) && palabCorrecta[j] == '_')
149                  palabCorrecta[j] = letra;
150              System.out.print(palabCorrecta[j] + " ");
151          }
152      }

```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 5</p>

Esta es la parte final del *main*, que verifica si ganaste o perdiste. El primer condicional se encarga de comprobar si la cadena de caracteres es igual a la palabra elegida. Si es correcto, emite un mensaje de victoria y sale del *while* con un contador menor a 7. El último condicional verifica que, si el contador es igual a 7, entonces has perdido.

```

93         // Verifica si el jugador ha ganado.
94         if (determinarSiGano(palabCorrecta, palSecreta)) {
95             System.out.println("¡GANASTE, FELICIDADES!");
96             break;
97         }
98     }
99     // Si el contador llega a 7, significa que el jugador ha perdido.
100    if (contador == 7)
101        System.out.println("Lo lamento, usted perdió.");
102 }

```

Este método verifica si la cadena de caracteres es igual a la palabra elegida, devolviendo *true* en caso de ser cierto; de lo contrario, devuelve *false*.

```

154     // Determina si el jugador ha ganado comparando la palabra formada con la secreta.
155    public static boolean determinarSiGano(char[] palabCorrecta, String palabra) {
156        String palabElegida = "";
157        for (char n : palabCorrecta)
158            palabElegida = palabElegida + n;
159        return (palabElegida.equals(palabra));
160    }
161 }

```

## COMMITTS:

```
commit e0a6de5e35420752d757fcba54f502acbc5780f (HEAD -> main, origin/main)
```

```
Author: Calihn1 <ehilacondob@unsa.edu.pe>
```

```
Date: Fri Sep 27 15:41:02 2024 -0500
```

Laboratorio2

```
commit 895fee80480bcf5b4c38d3559360475ae1f6e320
```

```
Author: Calihn1 <ehilacondob@unsa.edu.pe>
```

```
Date: Fri Sep 27 13:43:28 2024 -0500
```

Agregar el proyecto

```
emanu@DESKTOP-LT66QVT MINGW64 /e/LABORATORIO FP2/Laboratorio2 (main)
```

El primer *commit* subido fue el que avancé en clase, el cual ya tenía desarrollada la estructura de las funciones de acuerdo con el ejercicio. Aquí solo solucioné los problemas que tenían algunas funciones o faltaban por completar, como el algoritmo de letraEnPalabraSecreta, que aún necesitaba código para verificar si la letra ingresada por el usuario pertenece a la palabra en juego, y el método mostrarBlancosActualizados, que también faltaba terminar.

El segundo *commit* fue la creación del método para verificar si el usuario había ganado el juego o, en caso contrario, mostrar el mensaje de que perdió. Actualicé tanto el *main* con el último condicional creado dentro del *while*, como el segundo condicional que verifica si el contador llegó a 7.

## II. PRUEBAS

```

+---+
|   |
+---+

=====

- - - - -

Ingrese letra: a
+---+
|   |
0   |
+---+

=====

Ingrese letra: j
_ _ j _ _ _
Ingrese letra: p
+---+
|   |
0   |
|   |
+---+

=====

Ingrese letra: c
+---+
|   |
0   |
/|  |
+---+

=====

Ingrese letra: e
_ _ j e _ _ _
Ingrese letra: o
o _ j e _ o _
Ingrese letra: i
+---+
|   |
0   |
/|\  |
+---+

=====

Ingrese letra: u
+---+
|   |
0   |
/|\  |
/   |
+---+

=====

Ingrese letra: b
o b j e _ o _
Ingrese letra: t
o b j e t o _
Ingrese letra: s
o b j e t o s ¡GANASTE, FELICIDADES!

```

Como podemos ver, el programa funciona tal como está en el código. Si la letra no pertenece a la palabra, se genera una imagen que muestra el avance del ahorcado, mientras que, si aciertas, los guiones se van actualizando con las letras correspondientes, lo que al final te ayuda a adivinar la palabra.

```

Ingrese letra: a
_ a _ a
Ingrese letra: j
j a _ a
Ingrese letra: x
+---+
|   |
o   |
/|\  |
/   |

=====

Ingrese letra: c
| +---+
|   |
o   |
/|\  |
/|\  |

=====
Lo lamento, usted perdió.

```

Esta imagen demuestra que, al llegar a completar el ahorcado, terminas perdiendo, como indica el código.

## CUESTIONARIO:

### 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 método `ingreseLetra()` tiene un "for" que verifica si la el valor ingresado verdaderamente es un letra de a-z, en caso no sea cierto se reinicia hasta que ingrese una letra y la función recién vuelve.*

- *El juego supone que el usuario no ingresa una letra ingresada previamente*

*Tal y como indica se debe suponer, pero igualmente en la función `mostrarBlancosActualizados`, verifica si la letra ingresada ya esta en el juego y si lo esta ya lo toma para actualizar la cadena.*

- *El método `ingreseLetra()` debe ser modificado para incluir las consideraciones de validación*

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 8</p>

- *Puede crear métodos adicionales.*

*Se creó solo el método `determinoSiGano`.*

## CONCLUSIONES

Es sorprendente cómo se pueden crear juegos con las funciones básicas de Java. Además, la creación de métodos es fundamental para un desarrollo correcto del código, permitiendo así evitar revisar todo el método main cada vez que algo falle.

## METODOLOGÍA DE TRABAJO

*Primero entender el problema que plantea el ejercicio.*

*-Diseñar el algoritmo más corto y eficiente para solucionar el problema que tenían algunos métodos.*



*-Escribir el código en el lenguaje de programación java para ejecutar el algoritmo.*

*-Recibir los valores esperados del código realizado.*

## REFERENCIAS Y BIBLIOGRAFÍA

*En este caso no use ninguna referencia para realizar el código.*



	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLE-001	<b>Página:</b> 9

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

Contenido y demostración		Puntos	Checklis t	Estudiant e	Profeso r
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	3	
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 están 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	XX	2	

TOTAL	20	17		
-------	----	----	--	--

Tabla 2: Rúbrica para contenido del Informe y demostración