INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	*Fundamentos de la programación II*				
TÍTULO DE LA PRÁCTICA:	*Arreglos Estándar*				
NÚMERO DE PRÁCTICA:	*02*	**AÑO LECTIVO:**	*2024*	**NRO. SEMESTRE:**	*02*
FECHA DE PRESENTACIÓN	*29/09/2024*	**HORA DE PRESENTACIÓN**	*17/30/00*		
INTEGRANTE (s) *José León Enrique Hatches Curo*				**NOTA (0-20)**	*Nota colocada por el docente*
DOCENTE(s): *Ing. Lino Pinto Oppe*					

RESULTADOS Y PRUEBAS

I. EJERCICIOS RESUELTOS:

El estudiante coloca la evidencia de los ejercicios propuestos realizados en la sesión de laboratorio, en el tiempo o duración indicado por el docente.
El docente debe colocar la retroalimentación por cada ejercicio que el estudiante/grupo ha presentado.

EL 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.*

Repositorio en GitHub: LEON_HATCHES_LABORATORIO_2/LABORATORIO 02/src/Enunciado01.java at main · LeonHatches/LEON_HATCHES_LABORATORIO_2 (github.com)

- *Primero, en esta parte del código se inicializan las variables para el funcionamiento del juego.*

```
// Version del ahorcado entregado al 70%, ahora 100%

import java.util.Scanner;
public class Enunciado01
{
    public static void main (String [] args)
    {
        String ahor1 = "  +---+  \n"+
                       "  |   |  \n" +
                       "      |  \n" +
                       "      |  \n" +
                       "      |  \n" +
                       "      |  \n" +
                       "========= ";

        String ahor2 = "  +---+    \n"+
                       "  |   |    \n"+
                       "  O   |    \n"+
                       "      |    \n"+
                       "      |    \n"+
                       "      |    \n"+
                       "=========";

        String ahor3 = "  +---+    \n"+
                       "  |   |    \n"+
                       "  O   |    \n"+
                       "  |   |    \n"+
                       "      |    \n"+
                       "      |    \n"+
                       "=========";

        String ahor4 = "  +---+    \n"+
                       "  |   |    \n"+
                       "  O   |    \n"+
                       " /|   |    \n"+
                       "      |    \n"+
                       "      |    \n"+
                       "=========";

        String ahor5 = "  +---+    \n"+
                       "  |   |    \n"+
                       "  O   |    \n"+
                       " /|\\  |    \n"+
                       "      |    \n"+
                       "      |    \n"+
                    "=========";

        String ahor6 = "  +---+    \n"+
                       "  |   |    \n"+
                       "  O   |    \n"+
                       " /|\\  |    \n"+
                       " /    |    \n"+
                       "      |    \n"+
                       "=========";

        String ahor7 = "  +---+    \n"+
                       "  |   |    \n"+
                       "  O   |    \n"+
                       " /|\\  |    \n"+
                       " / \\  |    \n"+
                       "      |    \n"+
                       "=========";

        int contador = 0;
        String letra, palSecreta;
        String [] figuras  = {ahor1, ahor2, ahor3, ahor4, ahor5, ahor6, ahor7};
        String [] palabras = {"programacion", "java", "indentacion", "clases"   ,
                              "objetos"     , "desarrollador"      , "pruebas"};
```

- *Luego, en esta parte recogemos la palabra secreta del arreglo inicializado mediante un método al azar de palabras, para después, darle valores en MAYÚS. Continuando, se crea un arreglo de espacios en blanco "_"; este será inicializado con un método, para luego ser mostrado con el próximo.*

```
// Palabra secreta en mayusculas
palSecreta = getPalabraSecreta(palabras).toUpperCase();

// Arreglo para espacios en blancos para actualizar
String [] blancos = new String [palSecreta.length()];

// Figura inicial
System.out.println ( figuras[0] );

// INICIALIZA Y MUESTRA LOS BLANCOS
inicializarBlancos (blancos);
mostrarBlancos     (blancos);
System.out.println ("\n");
```

- *Después, se hace una sentencia repetitiva con un contador que comienza a partir del "0" para mostrar correctamente la figura y no contarla como errónea, para luego ingresar la letra mediante un método y así pasarla por una sentencia condicional con un verificador si contiene la palabra; en dado caso si, se actualiza el arreglo de espacios en blancos o en dado caso no, se suma uno en el contador para cambiar la figura. Por último, un verificador si existen espacios en blancos, así decide si gana o pierde.*

```
    while ( contador < 6 && !verificadorEspacios(blancos) )
    {
        // Entrada de letra
        letra = ingreseLetra();

        // Verificador de letra en palabra
        if ( letraEnPalabraSecreta  (letra, palSecreta) )
                blancosActualizados (letra, blancos, palSecreta);
        else
            contador++;

        // Muestra datos del juego
        System.out.println (figuras[contador]);
        mostrarBlancos (blancos);
    }

    // Verificador de victoria
    if ( verificadorEspacios(blancos) )
         System.out.println("¡Usted GANA el juego del ahorcado!");
    else
        System.out.println("Usted PIERDE el juego del ahorcado."
                           + "\nLa palabra era: "+palSecreta);
}
```

- *Este método se encarga de escoger la palabra secreta mediante un "Math.random()" obteniendo el índice y devolviendo una cadena del arreglo.*

```
public static String getPalabraSecreta (String [] lasPalabras)
{
    int ind, indiceMayor = lasPalabras.length;

    ind = (int) ( Math.random()*indiceMayor );

    return lasPalabras[ind];
 }
```

- *Este método inicializa el arreglo con "_".*

```
public static void inicializarBlancos (String [] blancos)
{
    for (int i = 0 ; i < blancos.length ; i++)
        blancos [i] = "_";
}
```

- *Este método muestra el arreglo de los blancos, también si están actualizados con las letras.*

```
public static void mostrarBlancos (String [] blancos)
{
    for (String espacio : blancos)
        System.out.print (espacio+" ");

    System.out.println();
}
```

- *Este método devuelve el ingreso de una letra, con una verificación si es dígito o no, con un método de Character, para luego devolver la letra respetando que sea de tipo cadena como el código base.*

```
public static String ingreseLetra ()
{
    Scanner sc = new Scanner(System.in);
    char laLetra;

    System.out.println ("Ingrese letra: ");
    laLetra = sc.next().toUpperCase().charAt(0);

    while ( Character.isDigit (laLetra) )
    {
        System.out.println("| CARACTER INVALIDO |\n");
        System.out.println("Ingrese letra: ");
        laLetra = sc.next().toUpperCase().charAt(0);
    }
    return Character.toString(laLetra);
}
```

- *Este método, verifica si la letra ingresada está en la palabra secreta, devolviendo un verdadero o falso.*

```
public static boolean letraEnPalabraSecreta (String letra, String palSecreta)
{
    for (int i = 0 ; i < palSecreta.length() ; i++)
    {
        if ( letra.equals (palSecreta.substring(i, i+1)) )
            return true;
    }
    return false;
}
```

- *Este método se encarga de actualizar los espacios en blanco con la letra asignada.*

```
public static void blancosActualizados (String letra, String [] blancos, String palSecreta)
{
    System.out.println("\nPROCESANDO.....\n");

    for (int i = 0 ; i < palSecreta.length() ; i++)
    {
        if ( letra.equals ( palSecreta.substring (i, i+1) ) )
            blancos[i] = letra;
    }
}
```

- *Este método verifica si el arreglo que fue actualizado con las 6 iteraciones sigue teniendo espacios en blanco "_", en cualquier caso, devolverá un booleano.*

```
public static boolean verificadorEspacios (String [] blancos)
{
    for (String espacio : blancos)
        if (espacio == "_")
            return false;

    return true;
}
```

II. PRUEBAS

EJECUCIÓN:

- *Esta es la parte inicial del juego del ahorcado, donde aparecerá la figura y los espacios en blanco.*

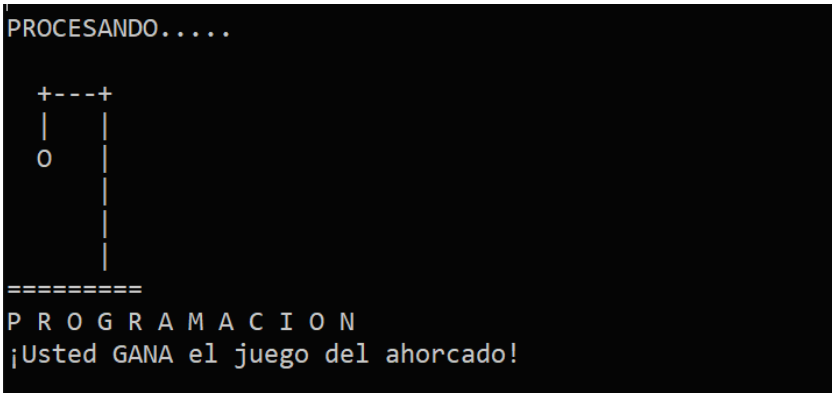
```
  +---+
  |   |
      |
      |
      |
      |
=========
_ _ _ _ _ _ _ _ _ _ _ _

Ingrese letra:
```

- *Luego de ingresar la letra, verificará si existe la letra en la palabra secreta para terminar mostrando la figura y los espacios en blanco actualizados.*

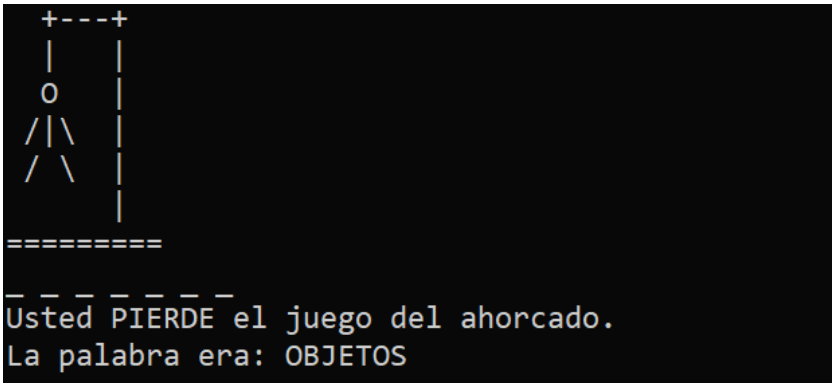
```
Ingrese letra:
a

PROCESANDO.....

  +---+
  |   |
      |
      |
      |
      |
=========
_ _ _ _ _ A _ A _ _ _ _
Ingrese letra:
```

- *Aquí se puede observar como los espacios en blanco se van actualizando sin perder las demás letras.*

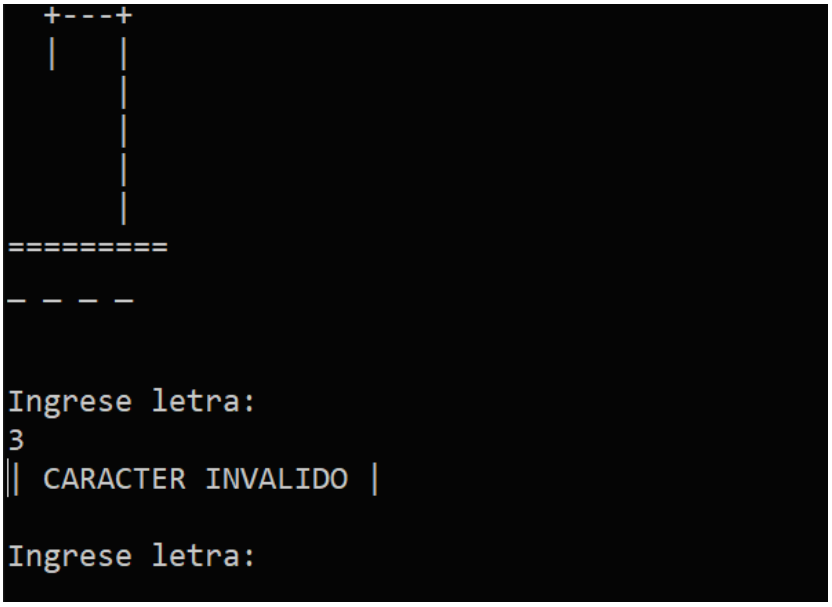
- *En esta captura se puede apreciar la pantalla cuando se consigue adivinar la palabra.*



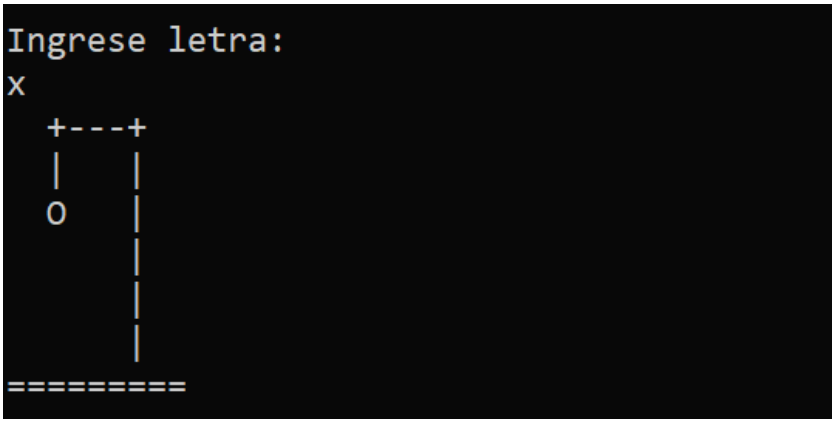
- *En la siguiente captura, se muestra la pantalla cuando se pierde y no se logra adivinar la palabra.*



- *Aquí se observa como rechaza números no deseados.*



- *En esta última captura, se observa si se ingresa un valor equivocado.*



III. CUESTIONARIO:

Colocar la evidencia de las respuestas realizadas al cuestionario enunciado en la guía práctica de laboratorio.

Preguntas:

¿Con qué valores comprobaste que tu práctica estuviera correcta?

- *Con valores de entrada tipo String (cadena), que son representadas por letras.*

¿Qué resultado esperabas obtener para cada valor de entrada?

- *Esperaba resultados de salida del arreglo que va actualizando las letras en cada iteración, como también un mensaje para saber si se gana o se pierde.*

¿Qué valor o comportamiento obtuviste para cada valor de entrada?

- *Obtuve los valores que deseaba en la respuesta anterior, que me devolviera lo necesario para el juego del ahorcado.*

CONCLUSIONES

En conclusión, para que el juego del ahorcado pueda tener un buen funcionamiento, el uso de arreglos será necesario de implementar para representar los espacios en blanco y su actualización, así mostremos gran optimización, accesibilidad y orden al momento de programar.
Así también, la importancia de los métodos al instante de trabajar y tener un código que tenga una facilidad de observación al momento de las correcciones del código.

METODOLOGÍA DE TRABAJO

Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el procedimiento/secuencia de pasos en forma general.

- *La secuencia que utilicé fue, primero, analizar el código base ya proporcionado y tratar de darle parte de la solución en mi mente. Segundo, me dediqué a armar la estructura de los algoritmos faltantes y los distintos métodos. Tercero, analizar errores y formas de optimizar el código para que se visualice de una manera entendible. Cuarto y último, compilar o correr el código para visualizar su funcionamiento o en dado caso no funcione, volver al paso 3 para corregir.*

REFERENCIAS Y BIBLIOGRAFÍA

Ninguna referencia externa.

RÚBRICA PARA EL CONTENIDO DEL INFORME Y DEMOSTRACIÓN

El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumplió 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 los ítems.

El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

Tabla 1: 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

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	1	
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 están dentro de los plazos de fecha de entrega establecidos.	2	X	1	
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		16	

Tabla 2: Rúbrica para contenido del Informe y demostración