
	UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

## INFORME DE LABORATORIO

### (formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programación 2 - Lab				
TÍTULO DE LA PRÁCTICA:	Arreglos Estandar				
NÚMERO DE PRÁCTICA:	02	AÑO LECTIVO:	2024	NRO. SEMESTRE:	02
FECHA DE PRESENTACIÓN	27/09/2024	HORA DE PRESENTACIÓN	21/52/23		
INTEGRANTE (s) Sergio Emilio Estrada Arce				NOTA (0-20)	Nota colocada por el docente
DOCENTE(s): Lino Pinto Oppa					

RESULTADOS Y PRUEBAS
I. EJERCICIOS RESUELTOS:

```



1 //Laboratorio Nro 2 - Ejercicio 1
2 //Autor: Sergio Estrada Arce
3 //Tiempo: 1h con 40m
4 package Ejercicios;
5 import java.util.*;
6 public class Ejercicio1 {
7     public static void main(String[] args) {
8         // Figuras del ahorcado en diferentes etapas según los errores
9         String ahor1 =
10             " +---+\n" +
11             " |  |\n" +
12             " |  |\n" +
13             " |  |\n" +
14             " |  |\n" +
15             " |  |\n" +
16             " =====\n";
17         String ahor2 =
18             " +---+\n" +
19             " |  |\n" +
20             " 0  |\n" +
21             " |  |\n" +
22             " |  |\n" +
23             " |  |\n" +
24             " =====\n";
25         String ahor3 =
26             " +---+\n" +
27             " |  |\n" +
28             " 0  |\n" +
29             " |  |\n" +
30             " |  |\n" +
31             " |  |\n" +
32             " =====\n";
33         String ahor4 =
34             " +---+\n" +
35             " |  |\n" +
36             " 0  |\n" +
37             " |  |\n" +
38             " /  |\n" +
39             " |  |\n" +
40             " =====\n";
41         String ahor5 =
42             " +---+\n" +
43             " |  |\n" +
44             " 0  |\n" +
45             " |  |\n" +
46             " / \ \  |\n" +
47             " |  |\n" +
48             " =====\n";
49         String ahor6 =
50             " +---+\n" +
51             " |  |\n" +
52             " 0  |\n" +
53             " / \  |\n" +
54             " / \ \  |\n" +
55             " |  |\n" +
56             " =====\n";

```

```

57 String ahor7 =
58     " +---+\n" +
59     " |   |\n" +
60     " 0   |\n" +
61     "/|\ \ |\n" +
62     "/ \ \ |\n" +
63     "   |\n" +
64     " =====\n";
65
66 String[] figuras = {ahor1, ahor2, ahor3, ahor4, ahor5, ahor6, ahor7};
67
68 int contador = 1; // Lleva la cuenta de los errores (máximo 6)
69 String letra;
70
71 // Palabras disponibles para el juego
72 String[] palabras = {"programacion", "java", "indentacion", "clases", "objetos", "desarrollador", "pruebas"};
73 String palabraSecreta = getPalabraSecreta(palabras); // Selección aleatoria
74
75 // La palabra a adivinar se representa inicialmente con guiones bajos
76 StringBuilder palabraAdivinada = new StringBuilder("_".repeat(palabraSecreta.length()));
77
78 System.out.println(figuras[0]); // Muestra el estado inicial del ahorcado
79 mostrarBlancos(palabraAdivinada); // Muestra la palabra oculta (con guiones)
80
81 System.out.println("\n");
82
83 // Bucle que sigue hasta que se adivine la palabra o se alcancen los 6 errores
84 while (contador <= 6 && !palabraCompletada(palabraAdivinada)) {
85     letra = ingreseLetra(); // Se obtiene la letra ingresada por el jugador
86
87     if (letraEnPalabraSecreta(letra, palabraSecreta)) {
88         // Actualiza la palabra mostrada si la letra es correcta
89         actualizarBlancos(letra, palabraSecreta, palabraAdivinada);
90         mostrarBlancos(palabraAdivinada);
91     } else {
92         // Muestra la siguiente etapa del ahorcado en caso de error
93         System.out.println(figuras[contador]);
94         contador++; // Se incrementa el número de errores
95     }
96 }
97
98 // Verifica si el jugador ganó o perdió, y muestra el mensaje correspondiente
99 if (palabraCompletada(palabraAdivinada)) {
100     System.out.println("¡Felicidades! Adivinaste la palabra: " + palabraSecreta);
101 } else {
102     System.out.println("Lo siento, perdiste. La palabra secreta era: " + palabraSecreta);
103 }
104
105 System.out.println("\n");
106 }
107
108 // Selecciona aleatoriamente una palabra del arreglo de palabras
109 public static String getPalabraSecreta(String[] lasPalabras) {
110     int indiceMayor = lasPalabras.length - 1;
111     int ind = (int) ((Math.random() * (indiceMayor + 1)));
112     return lasPalabras[ind];
113 }

```

	<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>

```

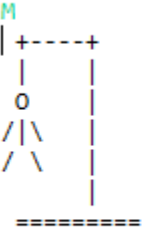
114
115 // Muestra la palabra oculta con guiones o letras adivinadas
116 public static void mostrarBlancos(StringBuilder palabraAdivinada) {
117     for (int i = 0; i < palabraAdivinada.length(); i++) {
118         System.out.print(palabraAdivinada.charAt(i) + " ");
119     }
120     System.out.println();
121 }
122
123 // Pide al usuario una letra y verifica que sea válida (una sola letra de 'a' a 'z')
124 public static String ingreseLetra() {
125     Scanner sc = new Scanner(System.in);
126     System.out.println("Ingrese una letra: ");
127     String laLetra = sc.next().toLowerCase();
128
129     // Asegura que solo se ingrese una letra válida
130     while (laLetra.length() != 1 || !laLetra.matches("[a-z]")) {
131         System.out.println("Entrada inválida. Ingrese solo una letra (a-z): ");
132         laLetra = sc.next().toLowerCase();
133     }
134     return laLetra;
135 }
136
137 // Verifica si la letra ingresada está en la palabra secreta
138 public static boolean letraEnPalabraSecreta(String letra, String palabraSecreta) {
139     return palabraSecreta.contains(letra); // Uso del método contains
140 }
141
142 // Actualiza los guiones bajos por la letra adivinada en la posición correcta
143 public static void actualizarBlancos(String letra, String palabraSecreta, StringBuilder palabraAdivinada) {
144     for (int i = 0; i < palabraSecreta.length(); i++) {
145         if (palabraSecreta.charAt(i) == letra.charAt(0)) {
146             palabraAdivinada.setCharAt(i, letra.charAt(0));
147         }
148     }
149 }
150
151 // Verifica si la palabra ya ha sido completamente adivinada
152 public static boolean palabraCompletada(StringBuilder palabraAdivinada) {
153     return !palabraAdivinada.toString().contains("_");
154 }
155 }
156
157
158

```

## II. PRUEBAS



Ingrese una letra:



Lo siento, perdiste. La palabra secreta era: java

SrEstrada / Laboratorios\_Estrada\_Arce

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Laboratorios\_Estrada\_Arce

Public

Pin

Unwatch

1

Fork

Star

0

main

1 Branch

0 Tags

Go to file

Add file

Code

SrEstrada

Update Sergio Emilio Estrada Arce.txt

2a672f · 25 minutes ago

9 Commits

FP2 lab

Update Sergio Emilio Estrada Arce.txt

25 minutes ago

README.md

Update README.md

yesterday

README

Laboratorios\_Estrada\_Arce

-Nombre: Sergio Estrada Arce -Laboratorio: Fundamentos de la Programación 2 - F

About

No description, website, or topics provided.

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

© 2024 GitHub, Inc.

Terms

Privacy

Security

Status

Docs

Contact

Manage cookies



Do not share my personal information

## Update Sergio Emilio Estrada Arce.txt

SrEstrada committed 39 minutes ago Verified

En este código aún no se pusieron los comentarios explicativos debido a que la mayor parte del trabajo fue hecho en clase y en casa solo se necesito 1 versión para completar el trabajo

main

	<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>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 7

```

... @@ -0,0 +1,132 @@
1  + //Laboratorio Nro 2 - Ejercicio 1
2  + //Autor: Sergio Estrada Arce
3  + //Tiempo: 1h con 40m
4  + package Ejercicios;
5  + import java.util.*;
6  + public class Ejercicio1 {
7  +     public static void main(String[] args) {
8  +         String ahor1 =
9  +             " +---+ \n" +
10 +             " |   | \n" +
11 +             "   | \n" +
12 +             "   | \n" +
13 +             "   | \n" +
14 +             "   | \n" +
15 +             " +---+ \n";
16 +         String ahor2 =
17 +             " +---+ \n" +
18 +             " |   | \n" +
19 +             "  O  | \n" +
20 +             "   | \n" +
21 +             "   | \n" +
22 +             "   | \n" +
23 +             " +---+ \n";
24 +         String ahor3 =
25 +             " +---+ \n" +
26 +             " |   | \n" +
27 +             "  O  | \n" +
28 +             " |   | \n" +
29 +             "   | \n" +
30 +             "   | \n" +
31 +             " +---+ \n";
32 +         String ahor4 =
33 +             " +---+ \n" +
34 +             " |   | \n" +
35 +             "  O  | \n" +
36 +             " |   | \n" +
37 +             "  /  | \n" +
38 +             "   | \n" +
39 +             " +---+ \n";
40 +         String ahor5 =
41 +             " +---+ \n" +
42 +             " |   | \n" +
43 +             "  O  | \n" +
44 +             " |   | \n" +
45 +             "  / \ \  | \n" +
46 +             "   | \n" +
47 +             " +---+ \n";



```

```

48 +     String ahor6 =
49 +         " +----+\n" +
50 +         " |   |\n" +
51 +         " O   |\n" +
52 +         " /|   |\n" +
53 +         " / \ \ |\n" +
54 +         "   |\n" +
55 +         " ----- \n";
56 +     String ahor7 =
57 +         " +----+\n" +
58 +         " |   |\n" +
59 +         " O   |\n" +
60 +         " /|\ \ |\n" +
61 +         " / \ \ |\n" +
62 +         "   |\n" +
63 +         " ----- \n";
64 +
65 +     String[] figuras = {ahor1, ahor2, ahor3, ahor4, ahor5, ahor6, ahor7};
66 +     int contador = 1;
67 +     String letra;
68 +     String[] palabras = {"programacion", "java", "indentacion", "clases", "objetos", "desarrollador", "pruebas"};
69 +     String palabraSecreta = getPalabraSecreta(palabras);
70 +     StringBuilder palabraAdivinada = new StringBuilder("_".repeat(palabraSecreta.length()));
71 +     System.out.println(figuras[0]);
72 +     mostrarBlancos(palabraAdivinada);
73 +     System.out.println("\n");
74 +     while (contador <= 6 && !palabraCompletada(palabraAdivinada)) {
75 +         letra = ingresoLetra();
76 +         if (letraEnPalabraSecreta(letra, palabraSecreta)) {
77 +             actualizarBlancos(letra, palabraSecreta, palabraAdivinada);
78 +             mostrarBlancos(palabraAdivinada);
79 +         } else {
80 +             System.out.println(figuras[contador]);
81 +             contador++;
82 +         }
83 +     }
84 +     if (palabraCompletada(palabraAdivinada)) {
85 +         System.out.println("¡Felicidades! Adivinaste la palabra: " + palabraSecreta);
86 +     } else {
87 +         System.out.println("Lo siento, perdiste. La palabra secreta era: " + palabraSecreta);
88 +     }
89 +     System.out.println("\n");
90 + }
91 + public static String getPalabraSecreta(String[] lasPalabras) {
92 +     int indiceMayor = lasPalabras.length - 1;
93 +     int indiceMenor = 0;

```



	<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> 9</p>

```

94 +         int ind = (int) ((Math.random() * (indiceMayor - indiceMenor + 1)) + indiceMenor);
95 +         return lasPalabras[ind];
96 +     }
97 +     public static void mostrarBlancos(StringBuilder palabraAdivinada) {
98 +         for (int i = 0; i < palabraAdivinada.length(); i++) {
99 +             System.out.print(palabraAdivinada.charAt(i) + " ");
100 +         }
101 +         System.out.println();
102 +     }
103 +     public static String ingreseLetra() {
104 +         String laLetra;
105 +         Scanner sc = new Scanner(System.in);
106 +         System.out.println("Ingrese una letra: ");
107 +         laLetra = sc.next().toLowerCase();
108 +         while (laLetra.length() != 1 || !laLetra.matches("[a-z]")) {
109 +             System.out.println("Entrada inválida. Ingrese solo una letra (a-z): ");
110 +             laLetra = sc.next().toLowerCase();
111 +         }
112 +         return laLetra;
113 +     }
114 +     public static boolean letraEnPalabraSecreta(String letra, String palabraSecreta) {
115 +         return palabraSecreta.contains(letra);
116 +     }
117 +     public static void actualizarBlancos(String letra, String palabraSecreta, StringBuilder palabraAdivinada) {
118 +         for (int i = 0; i < palabraSecreta.length(); i++) {
119 +             if (palabraSecreta.charAt(i) == letra.charAt(0)) {
120 +                 palabraAdivinada.setCharAt(i, letra.charAt(0));
121 +             }
122 +         }
123 +     }
124 +     public static boolean palabraCompletada(StringBuilder palabraAdivinada) {
125 +         for (int i = 0; i < palabraAdivinada.length(); i++) {
126 +             if (palabraAdivinada.charAt(i) == '_') {
127 +                 return false;
128 +             }
129 +         }
130 +         return true;
131 +     }
132 + }

```

Update Sergio Emilio Estrada Arce.txt

Sfestrada committed 27 minutos ago [Verified](#)

En esta versión ya puse los comentarios explicativos sobre las partes más importantes del código.

Selección de la palabra (líneas 32-33): Se elige una palabra al azar y se prepara la variable palabraAdivinada con los guiones bajos.

Ejecución del juego (líneas 38-48): Aquí es donde se desarrolla el ciclo principal del juego. Se repite mientras el jugador siga teniendo intentos y no haya adivinado la palabra. Dependiendo de si la letra está o no en la palabra, se actualiza el dibujo del ahorcado o la palabra mostrada.

Resultado final (líneas 49-55): Cuando se sale del ciclo, se determina si el jugador ganó o perdió basándose en si la palabra está completa o si se alcanzaron los 6 errores.

1' main

```

1  - //Laboratorio Nro 2 - Ejercicio 1
2  - //Autor: Sergio Estrada Arce
3  - //Tiempo: 1h con 40m

4  1  package Ejercicios;
5  2  import java.util.*;
6  3  public class Ejercicio1 {
7  4      public static void main(String[] args) {
8      5  +      // Figuras del ahorcado en diferentes etapas según los errores
9  6      String ahor1 =
10 7          " +----+\n" +
11 8          " |   |\n" +

@@ -61,72 +59,94 @@ public class Ejercicio1 {
61 59          "/ \ \   |\n" +
62 60          "      |\n" +
63 61          " ----- \n";
64  -
65 62  +
66 63      String[] figuras = {ahor1, ahor2, ahor3, ahor4, ahor5, ahor6, ahor7};
67 64  -
68 65  +      int contador = 1;
69 66  +
70 67  +      int contador = 1; // lleva la cuenta de los errores (máximo 6)
71 68  +
72 69  +      String letra;
73 70  +
74 71  +      // Palabras disponibles para el juego
75 72  +
76 73  +      String[] palabras = {"programacion", "java", "indentacion", "clases", "objetos", "desarrollador", "pruebas"};
77 74  -
78 75  +      String palabraSecreta = getPalabraSecreta(palabras);
79 76  +
80 77  +      String palabraSecreta = getPalabraSecreta(palabras); // Selección aleatoria
81 78  +
82 79  +      // La palabra a adivinar se representa inicialmente con guiones bajos
83 80  +
84 81  +      StringBuilder palabraAdivinada = new StringBuilder("_".repeat(palabraSecreta.length()));
85 82  -
86 83  +      System.out.println(figuras[0]);
87 84  -
88 85  +      mostrarBlancos(palabraAdivinada);
89 86  +
90 87  +
91 88  +      System.out.println(figuras[0]); // Muestra el estado inicial del ahorcado
92 89  +      mostrarBlancos(palabraAdivinada); // Muestra la palabra oculta (con guiones)
93 90  +
94 91  +
95 92  +      System.out.println("\n");
96 93  -
97 94  +      while (contador <= 6 && !palabraCompleta(palabraAdivinada)) {
98 95  -
99 96  +          letra = ingreseLetra();

```

```

79 +
80 + // Bucle que sigue hasta que se adivine la palabra o se alcancen los 6 errores
81 + while (contador <= 6 && !palabraCompletada(palabraAdivinada)) {
82 +     letra = ingreseLetra(); // Se obtiene la letra ingresada por el jugador
83 +
76 84     if (letraEnPalabraSecreta(letra, palabraSecreta)) {
85 + // Actualiza la palabra mostrada si la letra es correcta
77 86     actualizarBlancos(letra, palabraSecreta, palabraAdivinada);
78 87     mostrarBlancos(palabraAdivinada);
79 88     } else {
89 + // Muestra la siguiente etapa del ahorcado en caso de error
80 90     System.out.println(figuras[contador]);
81 -     contador++;
91 +     contador++; // Se incrementa el número de errores
82 92     }
83 93     }
94 +
95 + // Verifica si el jugador ganó o perdió, y muestra el mensaje correspondiente
84 96     if (palabraCompletada(palabraAdivinada)) {
85 97         System.out.println("¡Felicidades! Adivinaste la palabra: " + palabraSecreta);
86 98     } else {
87 99         System.out.println("Lo siento, perdiste. La palabra secreta era: " + palabraSecreta);
88 100     }
101 +
89 102     System.out.println("\n");
90 103     }
104 +
105 + // Selecciona aleatoriamente una palabra del arreglo de palabras
91 106     public static String getPalabraSecreta(String[] lasPalabras) {
92 107         int indiceMayor = lasPalabras.length - 1;
93 -         int indiceMenor = 0;
94 -         int ind = (int) ((Math.random() * (indiceMayor - indiceMenor + 1)) + indiceMenor);
108 +         int ind = (int) ((Math.random() * (indiceMayor + 1)));
95 109         return lasPalabras[ind];
96 110     }
111 +
112 + // Muestra la palabra oculta con guiones o letras adivinadas
97 113     public static void mostrarBlancos(StringBuilder palabraAdivinada) {
98 114         for (int i = 0; i < palabraAdivinada.length(); i++) {
99 115             System.out.print(palabraAdivinada.charAt(i) + " ");
100 116         }
101 117         System.out.println();
102 118     }
119 +
120 + // Pide al usuario una letra y verifica que sea válida (una sola letra de 'a' a 'z')
103 121     public static String ingreseLetra() {
104 -         String laLetra;
105 122         Scanner sc = new Scanner(System.in);
106 123         System.out.println("Ingresa una letra: ");
107 -         laLetra = sc.next().toLowerCase();

```

```

108 - while (laLetra.length() != 1 || !laLetra.matches("[a-z]")) {
124 +     String laLetra = sc.next().toLowerCase();
125 +
126 +     // Asegura que solo se ingrese una letra válida
127 +     while (laLetra.length() != 1 || !laLetra.matches("[a-z]")) {
109 128         System.out.println("Entrada inválida. Ingrese solo una letra (a-z): ");
110 129         laLetra = sc.next().toLowerCase();
111 130     }
112 131     return laLetra;
113 132 }

133 +
134 + // Verifica si la letra ingresada está en la palabra secreta
114 135 public static boolean letraEnPalabraSecreta(String letra, String palabraSecreta) {
115 -     return palabraSecreta.contains(letra);
136 +     return palabraSecreta.contains(letra); // Uso del método contains
116 137 }

138 +
139 + // Actualiza los guiones bajos por la letra adivinada en la posición correcta
117 140 public static void actualizarBlancos(String letra, String palabraSecreta, StringBuilder palabraAdivinada) {
118 141     for (int i = 0; i < palabraSecreta.length(); i++) {
119 142         if (palabraSecreta.charAt(i) == letra.charAt(0)) {
120 143             palabraAdivinada.setCharAt(i, letra.charAt(0));
121 144         }
122 145     }
123 146 }



147 +
148 + // Verifica si la palabra ya ha sido completamente adivinada
124 149 public static boolean palabraCompletada(StringBuilder palabraAdivinada) {
125 -     for (int i = 0; i < palabraAdivinada.length(); i++) {
126 -         if (palabraAdivinada.charAt(i) == '_') {
127 -             return false;
128 -         }
129 -     }
130 -     return true;
150 +     return !palabraAdivinada.toString().contains("_");
131 151 }
132 152 }

```

### III. CUESTIONARIO:

-----

## CONCLUSIONES

	<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> 13



Gracias a esta actividad me pude dar cuenta de la utilidad y funcionalidad de los métodos, además de que siempre se puede mejorar un código con el método de prueba y error, no siempre es un camino recto para encontrar la respuesta y lo más importante que yo destaco es el hecho de que uno tiene que entender los código ajenos, no solo el suyo.

### METODOLOGÍA DE TRABAJO

*En primer lugar me puse a analizar el código que se nos otorgo, luego de ello identifique lo más fácil de arreglar para ir paso por paso, de ahí me imaginé como debería imprimirse y que era necesario para ello y finalmente me puse manos a la obra para culminar el trabajo*

### REFERENCIAS Y BIBLIOGRAFÍA

“GitHub - SrEstrada/Laboratorios\_Estrada\_Arce”. GitHub. Accedido el 27 de septiembre de 2024. [En línea]. Disponible: [https://github.com/SrEstrada/Laboratorios\\_Estrada\\_Arce/tree/main](https://github.com/SrEstrada/Laboratorios_Estrada_Arce/tree/main)

	<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> 14



### RUBRICA 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 lo ítems.

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

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	1.5	
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	X	2.5	
TOTAL		20		16	

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p><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> 15</p>