

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

INFORME DE LABORATORIO

N° 02

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programación 2				
TÍTULO DE LA PRÁCTICA:	Arreglos Estándar				
NÚMERO DE PRÁCTICA:	02	AÑO LECTIVO:	2024-B	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	27/09/2024	HORA DE PRESENTACIÓN	20:30:05		
INTEGRANTE (s) Subia Huaicane Edson Fabricio				NOTA (0-20)	Nota colocada por el docente
DOCENTE(s): Lino José Pinto Oppe					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <p>Actividad 1JUEGO DEL AHORCADO:</p> <p>En este ejercicio se le solicita a usted implementar el juego del ahorcado utilizando el código parcial que se le entrega. Deberá considerar que:</p> <ul style="list-style-type: none"> 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. <p>En este enlace se encuentra mi repositorio y los commits que realicé para la creación y/o mejora de este programa: https://github.com/Q3son/Edson_Subia_Juego-del-Ahorcado.git</p> <p>Mis COMMITS:</p> <ol style="list-style-type: none"> Este es el primer commit que realicé, es prácticamente la primera versión de mi código fuente que subí en GitHub.





Q3son

Create Juego del Ahorcado_v1.2.1

20c235e · yesterday

History

Esta es la primera versión de mi código en Java sobre el juego del Ahorcado, el cuál está trabajado actualmente en un 85% del total, (un 15% más de la versión original presentada a nosotros por parte del docente). En ella se han corregido las verificaciones del ingreso de caracteres para que en el juego, solo se permitan letras de la "A" a la "Z".

	<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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 2</p>

```


103     public static String ingreseLetra(){
104         String laLetra;
105         Scanner sc = new Scanner(System.in);
106         System.out.println("Ingrese letra: ");
107         laLetra = sc.next().toLowerCase();
108         while(laLetra.length() != 1 || laLetra.charAt(0) < 'a' || laLetra.charAt(0) > 'z'){
109             System.out.println("Ingrese letra: ");
110             laLetra = sc.next().toLowerCase();
111         }
112         return laLetra;
113     }

```

En este destaco la corrección del método para el ingreso de letras permitidas; por otro lado el cuerpo del programa aún no está pulido, puesto que, no es la versión final (por ende no puse comentarios).

2. Este es mi segundo commit más importante, puesto que aquí destaco el de mejoras visuales y sobre todo, la creación de nuevos métodos y mejora de los anteriormente presentados para una mejor optimización del programa, también destaco el uso de comentarios en las líneas de código para su mejor entendimiento al momento de ser utilizado. (Destaco más detalles en las siguientes imágenes). Esta es la v.2.0.5

Edson_Subia_Juego-del-Ahorcado / Juego del Ahorcado_v2.0.5


Q3son
Create Juego del Ahorcado_v2.0.5
4b96d3d · yesterday
History

Mejoras al código del juego del Ahorcado

Estructuración del código: He organizado el código en métodos específicos para cada funcionalidad. Esto no solo mejora la legibilidad, sino que también facilita el mantenimiento y la extensión del código en el futuro.

Comentarios explicativos: Agregué comentarios en cada parte clave del código para explicar la lógica detrás de mis decisiones. Esto es fundamental, ya que ayuda a cualquier lector (incluyéndome) a entender rápidamente el propósito de cada sección sin necesidad de analizar cada línea.



Validación de entradas: Implementé un método para validar las entradas del usuario. Esto asegura que solo se acepten letras y evita errores inesperados durante la ejecución del programa. Al manejar entradas incorrectas de manera controlada, mejoro la experiencia del usuario.

Contador de intentos: Agregué una variable para llevar un registro del número de intentos realizados. Esto no solo proporciona una referencia clara para el jugador, sino que también permite ofrecer mensajes motivadores al final del juego, como "¡Inténtalo de nuevo!" o "¡Buen trabajo!".

Manejo de errores: Implementé un sistema para manejar errores comunes, como intentos de adivinar letras que ya han sido utilizadas. Esto contribuye a una experiencia de juego más fluida y evita frustraciones innecesarias.

Mejoras visuales: Añadí algunos elementos visuales, como líneas que representan letras no adivinadas y un contador de intentos restantes. Esto hace que el juego sea más atractivo y fácil de seguir.

Opciones de reinicio: Incorporé una opción que permite a los jugadores reiniciar el juego sin necesidad de cerrar la aplicación. Esto ofrece una experiencia más amigable y conveniente.

	<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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 3</p>

```

public static void mostrarBlancosActualizados(String letra, String palSecreta, boolean[] letrasAdivinadas) {
    for (int i = 0; i < palSecreta.length(); i++) { // Itera sobre cada letra de la palabra secreta
        if (palSecreta.charAt(i) == letra.charAt(0)) { // Si la letra coincide con la letra ingresada
            letrasAdivinadas[i] = true; // Marca la letra como adivinada
        }
    }
    // Muestra el estado actual de la palabra
    for (int i = 0; i < palSecreta.length(); i++) {
        if (letrasAdivinadas[i]) {
            System.out.print(palSecreta.charAt(i) + " "); // Muestra la letra adivinada
        } else {
            System.out.print("_ "); // Muestra un espacio en blanco si no ha sido adivinada
        }
    }
    System.out.println(); // Salto de línea al final
}

public static boolean palabraCompletada(boolean[] letrasAdivinadas) {
    for (boolean letraAdivinada : letrasAdivinadas) { // Verifica si todas las letras han sido adivinadas
        if (!letraAdivinada) { // Si alguna letra no ha sido adivinada
            return false; // La palabra no está completa
        }
    }
    return true; // Todas las letras han sido adivinadas
}
}

```



3. Y por último, el Commit Nro. 3, en el cuál realicé una gran mejora en cuanto al ordenamiento visual, donde reorganicé las figuras tipo “ahor” y las integré en un mismo arreglo en tan solo un par de líneas, reduciendo así el espacio y Nro. De líneas de código. Cabe resaltar, que enumeré y mejoré mis comentarios para un mejor entendimiento y profesionalismo del programa.

65 // 18. Método para ingresar letra

80 // 20. Método para verificar si la letra está en la palabra secreta

85 // 21. Método para mostrar letras adivinadas

102 // 22. Método para verificar si la palabra ha sido completada

	<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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p style="text-align: center;">Aprobación: 2022/03/01</p>	<p style="text-align: center;">Código: GUIA-PRLE-001</p>	<p style="text-align: right;">Página: 4</p>

Edson_Subia_Juego-del-Ahorcado / Ahorcado_v3.0.0

Q3son Create Ahorcado_v3.0.0

9577e70 · yesterday

Cambios importantes, versión final:

- Añadido el código del juego del Ahorcado (versión 3.0.0).
- Incluidos comentarios esenciales enumerados según la rúbrica.
- Decidí reducir el número de líneas del código, metiendo los String 'ahor' en un arreglo de String con el nombre 'figuras'

Code Blame 111 lines (101 loc) · 5.2 KB Code 55% faster with GitHub Copilot

Raw Copy Download Edit

```

1 package Lab2;
2 import java.util.Scanner;
3 //Autor: Subia Huaicane Edson Fabricio
4 //Tiempo: 40 minutos
5 //Colaborador: Mi persona
6 //Version del ahorcado al 100% v.2.0.1
7 public class Ahorcado {
8     public static void main(String []args){
9         String[] figuras = {
10             " +---+ \n |   | \n |   | \n |   | \n |   | \n ===== ",
11             " +---+ \n |   | \n | 0 | \n |   | \n |   | \n |   | \n ===== ",
12             " +---+ \n |   | \n | 0 | \n |   | \n |   | \n |   | \n ===== ",
13             " +---+ \n |   | \n | 0 | \n | \n/ | \n |   | \n |   | \n ===== ",
14             " +---+ \n |   | \n | 0 | \n | \n/\ \ | \n |   | \n |   | \n ===== ",
15             " +---+ \n |   | \n | 0 | \n | \n/\ \ | \n | \n | \n | \n | \n ===== ",
16             " +---+ \n |   | \n | 0 | \n | \n/\ \ | \n | \n \ \ | \n |   | \n |   | \n ===== "
17         };
18     };

```



En la siguiente sección mostraré la versión final de mi código fuente del programa, trabajado en Eclipse IDE, en cada captura de pantalla se visualizarán líneas de código fundamentadas correctamente y un programa bien elaborado, y la respectiva ejecución del programa final. – Se ocuparon 111 líneas de código en total.

```

1 package Lab2;
2 import java.util.Scanner;
3 //Autor: Subia Huaicane Edson Fabricio
4 //Tiempo: 40 minutos
5 //Colaborador: Mi persona
6 //Version del ahorcado al 100% v.2.0.1
7 public class Ahorcado {
8     public static void main(String []args){
9         String[] figuras = {
10             " +---+ \n |   | \n |   | \n |   | \n |   | \n ===== ",
11             " +---+ \n |   | \n | 0 | \n |   | \n |   | \n |   | \n ===== ",
12             " +---+ \n |   | \n | 0 | \n |   | \n |   | \n |   | \n ===== ",
13             " +---+ \n |   | \n | 0 | \n | \n/ | \n |   | \n |   | \n ===== ",
14             " +---+ \n |   | \n | 0 | \n | \n/\ \ | \n |   | \n |   | \n ===== ",
15             " +---+ \n |   | \n | 0 | \n | \n/\ \ | \n | \n | \n | \n | \n ===== ",
16             " +---+ \n |   | \n | 0 | \n | \n/\ \ | \n | \n \ \ | \n |   | \n |   | \n ===== "
17         };
18     };
19     int contador = 0; // 2. Contador de figuras
20     String[] palabras = {"programacion", "java", "indentacion", "clases", "objetos", "desarrollador", "pruebas"}; // 3. Lista de palabras
21     String palSecreta = getPalabraSecreta(palabras); // 4. Obtener la palabra secreta
22     boolean[] letrasAdivinadas = new boolean[palSecreta.length()]; // 5. Arreglo para letras adivinadas
23     int turnosRestantes = 6; // 6. Número de turnos disponibles
24
25     // 8. Bucla principal del juego
26     while (turnosRestantes > 0 && !palabraCompleta(letrasAdivinadas)) {
27         String letra = ingreseLetra(); // 9. Solicitar letra al usuario
28         if (letraEnPalabraSecreta(letra, palSecreta)) {
29             mostrarBlancosActualizados(letra, palSecreta, letrasAdivinadas); // 10. Mostrar progreso
30         } else {
31             contador++; // 11. Incrementar contador para la figura del ahorcado
32             System.out.println(figuras[contador]); // 12. Mostrar siguiente figura
33             turnosRestantes--; // 13. Disminuir turnos restantes
34             System.out.println("Turnos restantes: " + turnosRestantes);
35         }
36         System.out.println("\n");
37     }
38 }

```

```
41     }
42
43     // 14. Indicar si ganó o perdió
44     if (palabraCompletada(letrasAdivinadas)) {
45         System.out.println("¡Felicidades! Has adivinado la palabra: " + palSecreta);
46     } else {
47         System.out.println("¡Has perdido! La palabra era: " + palSecreta);
48     }
49     System.out.println("Fin del juego."); // 15. Mensaje de fin de juego
50 }
51
52 // 16. Método para obtener la palabra secreta
53 public static String getPalabraSecreta(String[] lasPalabras) {
54     int ind = (int) (Math.random() * lasPalabras.length); // Selección de índice aleatorio
55     return lasPalabras[ind]; // Devuelve la palabra seleccionada
56 }
57
58 // 17. Método para mostrar espacios en blanco
59 public static void mostrarBlancos(String palabra) {
60     for (int i = 0; i < palabra.length(); i++)
61         System.out.print(" "); // Mostrar un espacio en blanco por cada letra
62     System.out.println(); // Salto de línea
63 }
64
65 // 18. Método para ingresar letra
66 public static String ingreseLetra() {
67     Scanner sc = new Scanner(System.in);
68     String laLetra;
69     System.out.print("Ingrese letra: ");
70     laLetra = sc.next().toLowerCase(); // Convertir a minúscula
71
72     // 19. Validar que solo se ingrese una letra
73     while (laLetra.length() != 1 || laLetra.charAt(0) < 'a' || laLetra.charAt(0) > 'z') {
74         System.out.print("Entrada inválida. Ingrese una letra: ");
75         laLetra = sc.next().toLowerCase();
76     }
77     return laLetra; // Devuelve la letra ingresada
78 }
79
80 // 20. Método para verificar si la letra está en la palabra secreta
81 public static boolean letraEnPalabraSecreta(String letra, String palSecreta) {
82     return palSecreta.contains(letra); // Devuelve "true" si la letra está en la palabra secreta
83 }
84
85 // 21. Método para mostrar letras adivinadas
86 public static void mostrarBlancosActualizados(String letra, String palSecreta, boolean[] letrasAdivinadas) {
87     for (int i = 0; i < palSecreta.length(); i++) {
88         if (palSecreta.charAt(i) == letra.charAt(0)) {
89             letrasAdivinadas[i] = true; // Marca la letra como adivinada
90         }
91     }
92     for (int i = 0; i < palSecreta.length(); i++) {
93         if (letrasAdivinadas[i]) {
94             System.out.print(palSecreta.charAt(i) + " "); // Muestra la letra adivinada
95         } else {
96             System.out.print("_ "); // Muestra un espacio en blanco
97         }
98     }
99     System.out.println(); // Salto de línea
100 }
101
102 // 22. Método para verificar si la palabra ha sido completada
103 public static boolean palabraCompletada(boolean[] letrasAdivinadas) {
104     for (boolean letraAdivinada : letrasAdivinadas) {
105         if (!letraAdivinada) {
106             return false; // La palabra no está completa
107         }
108     }
109     return true; // Todas las letras han sido adivinadas
110 }
111 }
```

	<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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 6</p>

EJECUCIÓN DEL PROGRAMA (v3.0.0): "JUEGO DEL AHORCADO"

```

C:\Program Files\Java\jdk-21\bin\java.exe (20 Sept 2024, 0:51:40)
+---+
|
|
=====
-----



Ingrese letra: s
+---+
|
0
|
|
=====
Turnos restantes: 5

Ingrese letra: d
+---+
|
0
|
|
|
=====
Turnos restantes: 4

Ingrese letra: g
-----g-----

Ingrese letra: h
+---+
|
0
/|
|
|
=====
Turnos restantes: 3

```

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

```

Ingrese letra: p
p _ _ g _ _ _ _ _ _ _

Ingrese letra: r
p r _ g r _ _ _ _ _ _

Ingrese letra: o
p r o g r _ _ _ _ _ o _

Ingrese letra: a
p r o g r a _ a _ _ o _

Ingrese letra: m
p r o g r a m a _ _ o _

Ingrese letra: n
p r o g r a m a _ _ o n

Ingrese letra: c
p r o g r a m a c _ o n

Ingrese letra: i
p r o g r a m a c i o n

¡Felicidades! Has adivinado la palabra: programacion
Fin del juego.

```

4. PRUEBAS



¿Con qué valores comprobaste que tu práctica estuviera correcta?

Comprobé la práctica utilizando diferentes palabras del arreglo predefinido como: "programacion", "java", "clases", entre otras. Además, ingresé letras válidas (como a, b, c) y algunas letras incorrectas para observar el comportamiento del juego.

¿Qué resultado esperabas obtener para cada valor de entrada?

Esperaba que el juego:

- Actualizara la figura del ahorcado cada vez que se ingresara una letra incorrecta.*
- Mostrara las letras correctas adivinadas por el jugador y las colocara en su lugar correspondiente dentro de la palabra secreta.*
- Terminara con un mensaje de victoria si se completaba la palabra o de derrota si se acababan los intentos.*

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

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

Los resultados obtenidos fueron los esperados:

- El juego manejó correctamente las entradas válidas y mostró los guiones bajos o las letras adivinadas correctamente.
- La figura del ahorcado se actualizó en cada intento fallido, pasando por cada etapa hasta llegar a la última figura.
- El juego terminó correctamente cuando se completó la palabra o se agotaron los turnos, mostrando los mensajes de victoria o derrota según el caso.

5. CUESTIONARIO:

¿Cuáles fueron los datos más importantes que consideró para realizar el código?



Los datos más importantes que consideré para realizar el código del juego de Ahorcado fueron los siguientes:

1. **Palabra secreta:** Este dato es fundamental ya que el objetivo del juego es adivinar la palabra oculta. Se selecciona de un arreglo predefinido de palabras.
2. **Figura del ahorcado:** Representada por varias etapas visuales almacenadas en cadenas de texto. Esta figura se va actualizando a medida que el jugador comete errores, lo cual es clave para dar retroalimentación visual sobre el progreso del juego.
3. **Turnos restantes:** El número de intentos que le quedan al jugador antes de perder. Este dato es importante para controlar el flujo del juego y saber cuándo debe finalizarse en caso de que el jugador no adivine la palabra.
4. **Letras ingresadas por el usuario:** Estas son validadas para comprobar si pertenecen a la palabra secreta o no. El control de estas letras es crucial para actualizar correctamente los guiones bajos con las letras adivinadas.
5. **Estado de las letras adivinadas:** Utilicé un arreglo de booleanos para llevar un seguimiento de qué letras ya han sido descubiertas por el jugador y cuáles aún están ocultas, permitiendo mostrar el estado parcial de la palabra.

CONCLUSIONES

Colocar las conclusiones, apreciaciones reflexivas, opiniones finales a cerca de los resultados obtenidos de la sesión de laboratorio.

El desarrollo del programa demuestra una estructura bien organizada y un enfoque interactivo que mejora la experiencia del usuario. Se implementó un manejo efectivo de errores, asegurando la robustez del sistema ante entradas inesperadas. Los resultados obtenidos fueron consistentes y esperados, lo que resalta la eficacia de la lógica programada. Este proyecto no solo permitió la aplicación práctica de conceptos de programación, sino

	<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;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 9</p>

que también facilitó el aprendizaje y la identificación de áreas de mejora, contribuyendo al crecimiento continuo en mis habilidades de desarrollo de software.

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.

- a) **Comprensión del problema:** En esta etapa, revisé cada una de las actividades propuestas, identificando cuidadosamente las restricciones y los objetivos a alcanzar.
- b) **Diseño del algoritmo:** Planifiqué la secuencia lógica necesaria para implementar la solución, aplicando los conocimientos adquiridos en Fundamentos de Programación I y II.
- c) **Codificación:** Procedí a implementar los programas solicitados, asegurándome de utilizar correctamente los arreglos y métodos.
- d) **Pruebas:** Realicé pruebas adicionales para verificar que el código funcionara de manera correcta con diferentes casos de prueba.

REFERENCIAS Y BIBLIOGRAFÍA

Colocar las referencias utilizadas para el desarrollo de la práctica en formato IEEE

M. W. Aedo López, *Fundamentos de programación I: Java Básico*, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, Jul. 2019. ISBN: 978-612-4337-55-0. 116 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 21 x 29.7 cm

[https://github.com/LINOPINTO2023/FundProg2/blob/main/entregaLaboratorio01/Hilacondo Emanuel LABORATORIO_01.pdf](https://github.com/LINOPINTO2023/FundProg2/blob/main/entregaLaboratorio01/Hilacondo_Emanuel_LABORATORIO_01.pdf)

https://github.com/Q3son/Edson_Subia_Juego-del-Ahorcado.git

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 con 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	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	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	
TOTAL		20	8	18	