Informe de Laboratorio 02

Tema: Arreglos estándar

Nota

Estudiante	Escuela	Asignatura
Christian Mestas Zegarra cmestasz@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 13 Setiembre 2023	Al 20 Setiembre 2023

1. Tarea

- **Actividad 1:** 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 Microsoft Windows 10 Pro 64 bits
- Visual Studio Code 1.82.2
- Java Development Kit 17.0.1
- Git 2.41.0.windows.1
- Windows PowerShell 5.1.19041.3031
- Cuenta en GitHub con el correo institucional.
- Arreglos estándar.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/cmestasz/fp2-23b.git`
- URL para el laboratorio 01 en el Repositorio GitHub.
- `https://github.com/cmestasz/fp2-23b/tree/main/fase01/lab01`

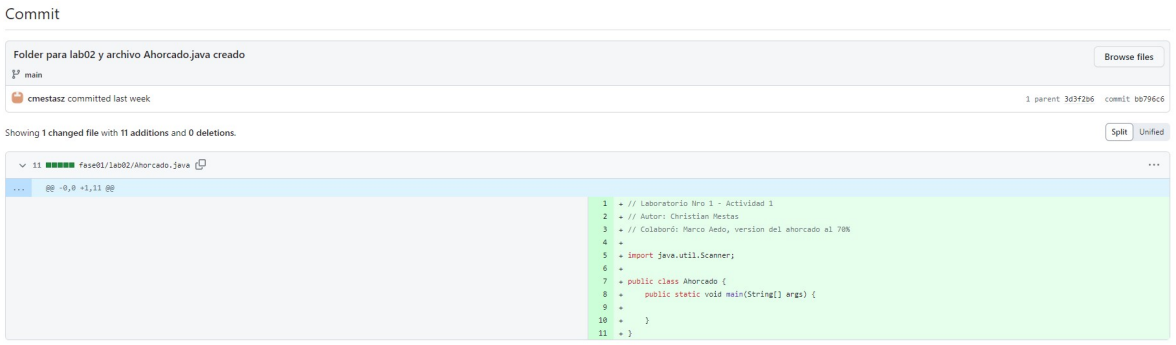
4. Actividades con el repositorio GitHub

Creando Ahorcado.java

```
$ mkdir lab02
$ cd lab02
$ code Ahorcado.java
```

Primer Commit / Ahorcado.java

```
$ git add Ahorcado.java
$ git commit -m "Folder para lab02 y archivo Ahorcado.java creado"
$ git push
```



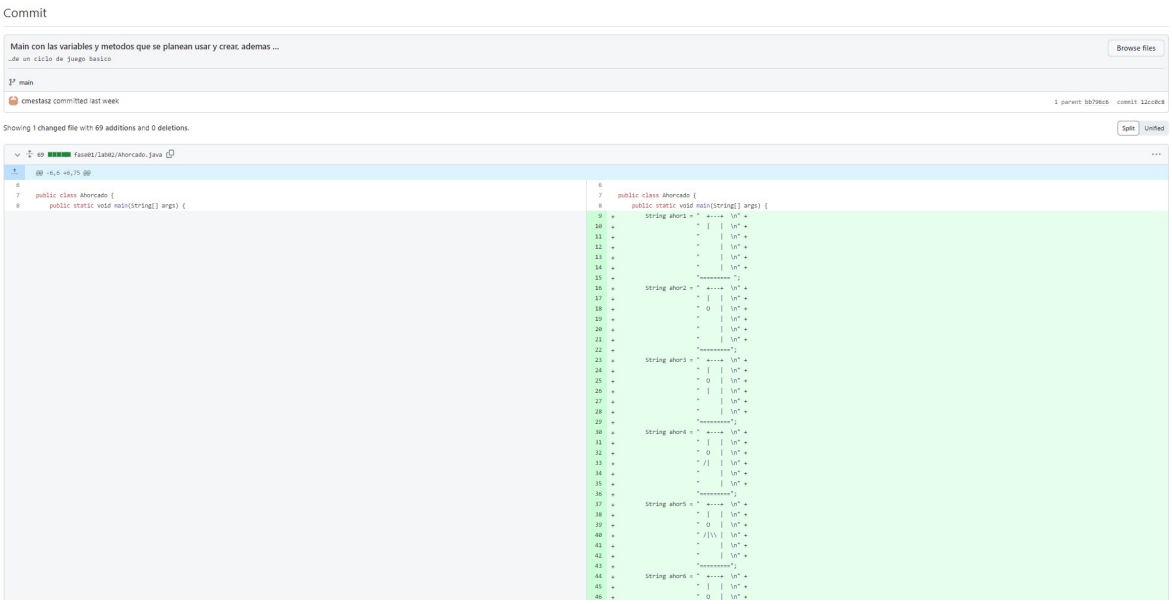
Primer Commit.

Actualizando Ahorcado.java

```
$ code Ahorcado.java
```

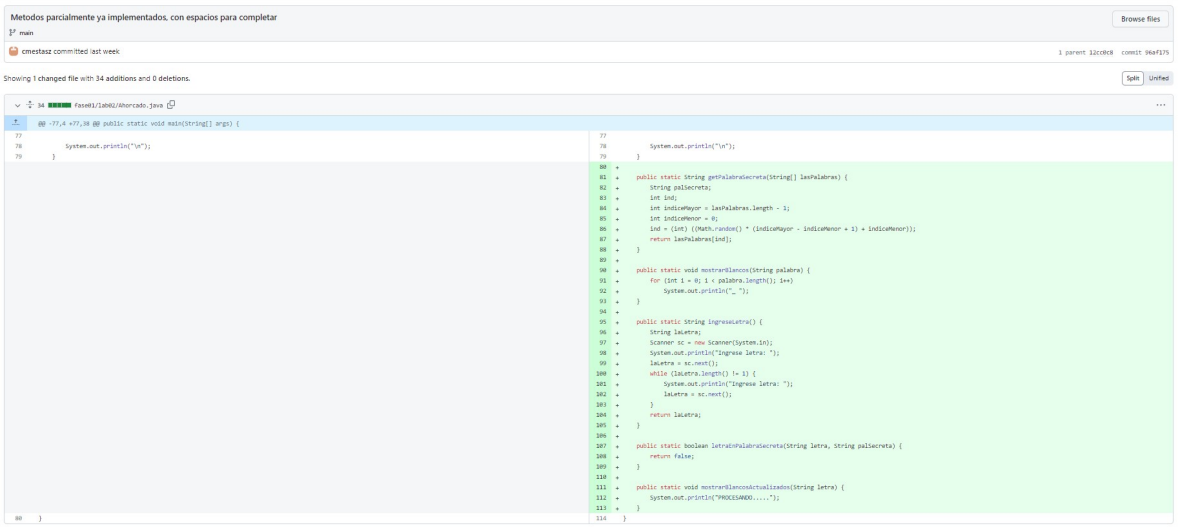
Segundo - Septimo Commit / Ahorcado.java

```
$ git add Ahorcado.java
$ git commit -m "Actividad 1"
$ code Ahorcado.java
$ git add Ahorcado.java
$ git commit -m "Actividad 2"
$ code Ahorcado.java
$ git add Ahorcado.java
$ git commit -m "Actividad 3"
$ code Ahorcado.java
$ git add Ahorcado.java
$ git commit -m "Actividad 4"
$ code Ahorcado.java
$ git add Ahorcado.java
$ git commit -m "Actividad 5"
$ git push
```



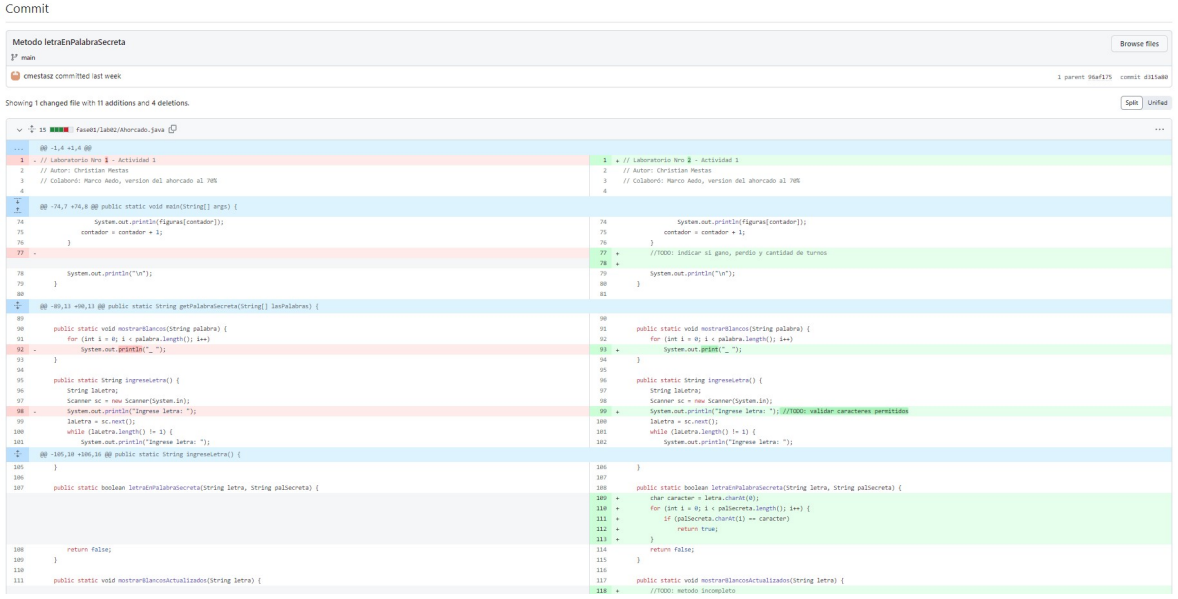
Segundo Commit.

Commit



Tercer Commit.

Commit



Cuarto Commit.

Commit

Metodo esLetraValida para validar entrada en metodo ingreseLetra

Browse files

main

cmestasz committed last week

1 parent d315a80 commit 6d7b71f

Showing 1 changed file with 9 additions and 2 deletions.

Split Unified

11 fase01/lab02/Ahorcado.java

```
@@ -96,9 +96,9 @@ public static void mostrarBlancos(String palabra) {
    public static String ingreseLetra() {
        String laLetra;
        Scanner sc = new Scanner(System.in);
-       System.out.println("Ingrese letra: "); //TODO: validar caracteres permitidos
        laLetra = sc.next();
-       while (laLetra.length() != 1) {
+       while (!esLetraValida(laLetra)) {
            System.out.println("Ingrese letra: ");
            laLetra = sc.next();
        }
@@ -118,4 +118,11 @@ public static void mostrarBlancosActualizados(String letra) {
        //TODO: metodo incompleto
        System.out.println("PROCESANDO.....");
    }
+
+   public static boolean esLetraValida(String letra) {
+       if (letra.length() != 1)
+           return false;
+       char caracter = letra.charAt(0);
+       return Character.isLetter(caracter);
+   }
}
```

Quinto Commit.

Commit

Metodo mostrarBlancosActualizados, con una manera de llevar la cuenta...

_ de las letras encontradas

Browse files

main

cmestasz committed last week

1 parent 6d7b71f commit 9a45951

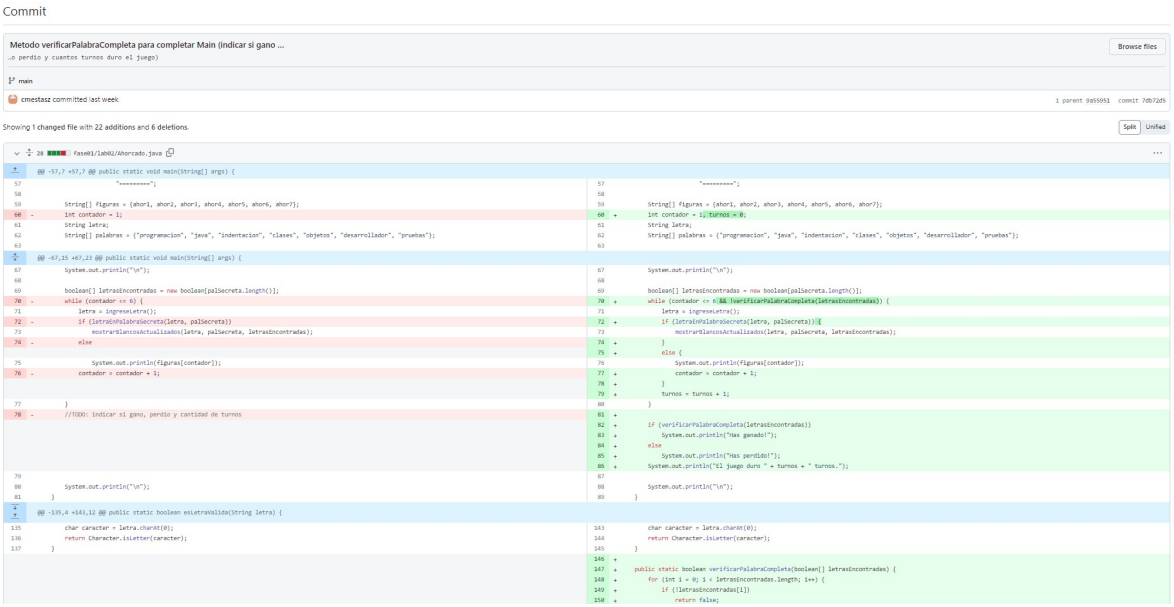
Showing 1 changed file with 16 additions and 6 deletions.

Split Unified

22 fase01/lab02/Ahorcado.java

```
@@ -66,10 +66,11 @@ public static void main(String[] args) {
        mostrarBlancos(palSecreta);
        System.out.println("\n");

+       boolean[] letrasEncontradas = new boolean[palSecreta.length()];
        while (contador <= 6) {
            letra = ingreseLetra();
            if (letraEnPalabraSecreta(letra, palSecreta))
-               mostrarBlancosActualizados(letra);
+               mostrarBlancosActualizados(letra, palSecreta, letrasEncontradas);
            else
                System.out.println(figuras[contador]);
            contador = contador + 1;
@@ -106,17 +107,26 @@ public static boolean letraEnPalabraSecreta(String letra, String palSecreta) {
    }
    return false;
}
-public static void mostrarBlancosActualizados(String letra) {
-    //TODO: metodo incompleto
-    System.out.println("PROCESANDO.....");
+public static void mostrarBlancosActualizados(String letra, String palSecreta, boolean[] letrasEncontradas) {
+    char letraIngresada = letra.charAt(0);
+    for (int i = 0; i < palSecreta.length(); i++) {
+        char letraActual = palSecreta.charAt(i);
+        if (letraActual == letraIngresada || letrasEncontradas[i]) {
+            System.out.print(letraActual + " ");
+            letrasEncontradas[i] = true;
+        } else {
+            System.out.print("_ ");
+        }
+    }
+    System.out.println("\n");
}

public static boolean esLetraValida(String letra) {
```

Sexto Commit.



Séptimo Commit.

5. Código desarrollado

Ahorcado.java

```
// Laboratorio Nro 2 - Actividad 1
// Autor: Christian Mestas
// Colaboro: Marco Aedo, version del ahorcado al 70%

import java.util.Scanner;

public class Ahorcado {
    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" +
```

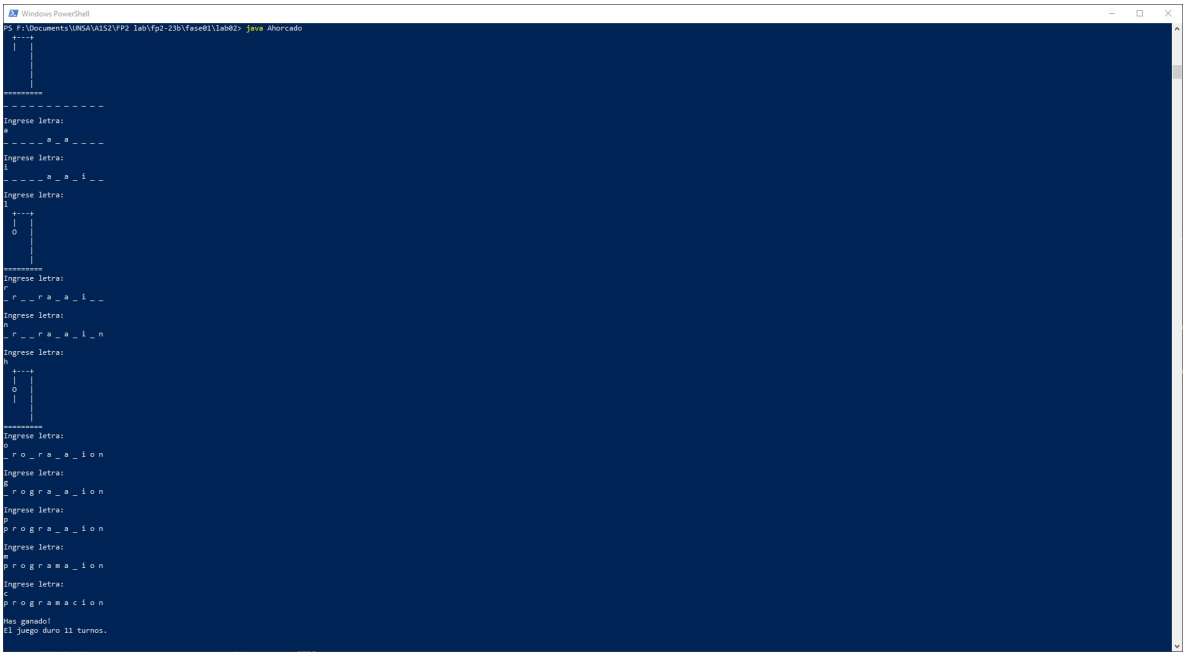
```
                    "  O   |  \n" +
                    " /|\\ |  \n" +
                    " / \\ |  \n" +
                    "     |  \n" +
                    "=========";

        String[] figuras = {ahor1, ahor2, ahor3, ahor4, ahor5, ahor6, ahor7};
        int contador = 1, turnos = 0;
        String letra;
        String[] palabras = {"programacion", "java", "indentacion", "clases", "objetos",
            "desarrollador", "pruebas"};

        String palSecreta = getPalabraSecreta(palabras);
        System.out.println(figuras[0]);
        mostrarBlancos(palSecreta);
        System.out.println("\n");

        boolean[] letrasEncontradas = new boolean[palSecreta.length()];
        while (contador <= 6 && !verificarPalabraCompleta(letrasEncontradas)) {
            letra = ingreseLetra();
            if (letraEnPalabraSecreta(letra, palSecreta)) {
                mostrarBlancosActualizados(letra, palSecreta, letrasEncontradas);
            }
            else {
                System.out.println(figuras[contador]);
                contador = contador + 1;
            }
            turnos = turnos + 1;
        }

        if (verificarPalabraCompleta(letrasEncontradas))
            System.out.println("Has ganado!");
        else
            System.out.println("Has perdido!");
        System.out.println("El juego duro " + turnos + " turnos.");

        System.out.println("\n");
    }

    public static String getPalabraSecreta(String[] lasPalabras) {
        String palSecreta;
        int ind;
        int indiceMayor = lasPalabras.length - 1;
        int indiceMenor = 0;
        ind = (int) ((Math.random() * (indiceMayor - indiceMenor + 1) + indiceMenor));
        return lasPalabras[ind];
    }

    public static void mostrarBlancos(String palabra) {
        for (int i = 0; i < palabra.length(); i++)
            System.out.print("_ ");
    }

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

```
108         System.out.println("Ingrese letra: ");
109         laLetra = sc.next();
110         while (!esLetraValida(laLetra)) {
111             System.out.println("Ingrese letra: ");
112             laLetra = sc.next();
113         }
114         return laLetra;
115     }
116 
117     public static boolean letraEnPalabraSecreta(String letra, String palSecreta) {
118         char letraIngresada = letra.charAt(0);
119         for (int i = 0; i < palSecreta.length(); i++) {
120             if (palSecreta.charAt(i) == letraIngresada)
121                 return true;
122         }
123         return false;
124     }
125 
126     public static void mostrarBlancosActualizados(String letra, String palSecreta, boolean[]
          letrasEncontradas) {
127         char letraIngresada = letra.charAt(0);
128         for (int i = 0; i < palSecreta.length(); i++) {
129             char letraActual = palSecreta.charAt(i);
130             if (letraActual == letraIngresada || letrasEncontradas[i]) {
131                 System.out.print(letraActual + " ");
132                 letrasEncontradas[i] = true;
133             } else {
134                 System.out.print("_ ");
135             }
136         }
137         System.out.println("\n");
138     }
139 
140     public static boolean esLetraValida(String letra) {
141         if (letra.length() != 1)
142             return false;
143         char caracter = letra.charAt(0);
144         return Character.isLetter(caracter);
145     }
146 
147     public static boolean verificarPalabraCompleta(boolean[] letrasEncontradas) {
148         for (int i = 0; i < letrasEncontradas.length; i++) {
149             if (!letrasEncontradas[i])
150                 return false;
151         }
152         return true;
153     }
154 }
```

- Se inicializan variables con los posibles estados del juego, posibles palabras secretas y un contador de turnos.
- getPalabraSecreta() regresa una palabra escogida al azar del arreglo palabras.
- mostrarBlancos() imprime la cantidad de letras que se tienen que descubrir.

- El juego inicia y continua hasta que el contador de errores sea 6 o la palabra este completa, verificado por el método verificarPalabraCompleta().
- En cada turno, ingreseLetra() recibe una letra del usuario, la que es verificada por el método esLetraValida(), letraEnPalabraSecreta() verifica si la letra es parte de la palabra secreta, y si lo es, mostrarBlancosActualizados() actualiza e imprime las letras encontradas y las que faltan encontrar.
- Si la letra no está en la palabra secreta, el contador de errores aumenta.
- Una vez el ciclo ha terminado, verificarPalabraCompleta() devuelve si la palabra esta completa y se imprime el resultado al usuario.
- Además se imprime la cantidad de turnos totales, los que aumentaban por cada iteración en el bucle anterior.

6. Ejecución del código



Ahorcado.java

7. Estructura de laboratorio 02

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab01/
|--- Ahorcado.java
|--- Informe.tex
|--- Informe.pdf
|--- img
   |--- logo_abet.png
   |--- logo_episunsa.png
   |--- logo_unsa.jpg
   |--- commit01.jpg
   |--- commit02.jpg
   |--- commit03.jpg
   |--- commit04.jpg
   |--- commit05.jpg
   |--- commit06.jpg
   |--- commit07.jpg
   |--- ejec01.jpg
```

8. Rúbricas

8.1. Entregable Informe

Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

8.2. 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 aprobatoria, siempre y cuando cumpla con todos los items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

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

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	4	
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	1.5	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2		0	
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		17	

9. Referencias

- Aedo, M. y Castro, E. (2021). FUNDAMENTOS DE PROGRAMACIÓN 2 - Tópicos de Programación Orientada a Objetos. Editorial UNSA.