

Informe de Laboratorio 03

Tema: Arreglos de Objetos

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
03	Arreglos de Objetos	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 18 Setiembre 2023	Al 25 Setiembre 2023

1. Tarea

- **Actividad 1:** Analice, complete y pruebe el Código de la clase DemoBatalla.
- **Actividad 2:** Solucionar la Actividad 4 de la Práctica 1 pero usando arreglo de objetos.
- **Actividad 3:** Solucionar la Actividad 5 de la Práctica 1 pero usando arreglos de objetos.

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 plantillas

```
$ mkdir lab03
$ cd lab03
$ code Nave.java
$ code DemoBatalla.java
$ code Soldado.java
$ code Actividad2.java
$ code Actividad3.java
```

Primer Commit / Plantillas

```
$ git add .
$ git commit -m "Plantillas de archivos necesarios para resolver las actividades del
laboratorio"
$ git push
```

Commit

Plantillas de archivos necesarios para resolver las actividades del L...
laboratorio

17 min
cmestias committed 1 hour ago

Showing 5 changed files with 133 additions and 0 deletions.

Filter changed files

- head/17a00
 - Actividad2.java
 - Actividad3.java
 - DemoBatalla.java
 - Nave.java
 - Soldado.java

Files changed:

- 0 FASE01/Lab01/Actividad2.java


```
1 // Laboratorio Nro 1 - Actividad 2
2 // Autor: Christian Mestas
3
4 public class Actividad2 {
5     public static void main(String[] args) {
6
7     }
8 }
```
- 0 FASE01/Lab01/Actividad3.java


```
1 // Laboratorio Nro 1 - Actividad 3
2 // Autor: Christian Mestas
3
4 public class Actividad3 {
5     public static void main(String[] args) {
6
7     }
8 }
```
- 09 FASE01/Lab01/DemoBatalla.java


```
1 // Laboratorio Nro 1 - Actividad 1
2 // Autor: Christian Mestas
3 // Solución: Naveos, Naves, clases DemoBatalla y Nave
4
5 import java.util.*;
6
7 public class DemoBatalla {
8     public static void main(String[] args) {
9         Nave[] naves = new Nave[10];
10        Scanner sc = new Scanner(System.in);
11        String nombre;
12        int fil, punt;
13        boolean est;
14        for (int i = 0; i < naves.length; i++) {
15            System.out.println("Nave " + (i + 1));
16            System.out.print("Nombre: ");
```

Primer Commit.

Actualizando Nave.java y DemoBatalla.java

```
$ code Nave.java
$ code DemoBatalla.java
```

Segundo - Sexto Commit / Nave.java y DemoBatalla.java

```
$ git add .
$ git commit -m "Metodo mostrarNaves() y metodo toString() de la clase Nave"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarPorNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarPorPuntos()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarMayorPuntos()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo desordenar()"
$ git push
```

Commit

Metodo mostrarNaves() y metodo toString() de la clase Nave

main

cmestasz committed 45 minutes ago

Showing 2 changed files with 10 additions and 1 deletion.

Filter changed files

ase07/ase03

Demodetalla.java

Nave.java

```

1 @@ -5,7 +5,7 @@
2
3 public class Demodetalla {
4     public static void main(String[] args) {
5         Nave[] naves = new Nave[10];
6         Scanner sc = new Scanner(System.in);
7         String nomb, col;
8         int f12, punt;
9
10        @@ -10,6 +10,8 @@ public static void main(String[] args) {
11
12        // Método para mostrar todas las naves
13        public static void mostrarNaves(Nave[] flota) {
14
15        }
16        // Método para mostrar todas las naves de un nombre que se pide por teclado
17        @@ -12,6 +14,7 @@ public static void mostrarNaves(Nave[] flota) {
18
19        // Método que devuelve la Nave con mayor número de Puntos
20        public static Nave mostrarMayorPuntos(Nave[] flota) {
21
22        }
23        // Crear un metodo que devuelva un nuevo arreglo de objetos con todos los
24        // objetos previamente ingresados
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

1 @@ -1,1 +1,5 @@
2
3 public class Nave {
4     private String nombre;
5     private int file;
6
7     @@ -11,1 +11,9 @@ public int getpuntos() {
8
9     return puntos;
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Segundo Commit.

Commit

Metodo mostrarPorNombre()

main

cmestasz committed 42 minutes ago

Showing 1 changed file with 8 additions and 0 deletions.

ase07/ase03

Demodetalla.java

```

45 @@ -45,6 +45,14 @@ public static void mostrarNaves(Nave[] flota) {
46
47 // Método para mostrar todas las naves de un nombre que se pide por teclado
48 public static void mostrarPorNombre(Nave[] flota) {
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

45 // Método para mostrar todas las naves de un nombre que se pide por teclado
46 public static void mostrarPorNombre(Nave[] flota) {
47     Scanner sc = new Scanner(System.in);
48     System.out.println("Ingresa el nombre a buscar:");
49     String nombre = sc.nextLine();
50     System.out.println("Naves con el nombre " + nombre + ":");
51     for (Nave nave : flota) {
52         if (nave.getNombre().equals(nombre)) {
53             System.out.println(nave.toString());
54         }
55     }
56
57 // Método para mostrar todas las naves con un número de puntos inferior o igual
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Tercer Commit.

Quinto Commit.

Commit

[illegible]

Sexto Commit.

Actualizando Soldado.java

```
$ code Soldado.java
```

Septimo Commit / Soldado.java

```
$ git add Soldado.java
$ git commit -m "Clase Soldado"
$ git push
```

Commit

Clase Soldado

main

cmestasz committed 17 minutes ago

1 parent 1b47e8a commit e406b59

Browse files

Showing 1 changed file with 27 additions and 1 deletion.

Split Unified

28

base01/1a003/Soldado.java

...

1

public class Soldado {

2

-

3

}

28

base01/1a003/Soldado.java

...

1

public class Soldado {

2

+ private String nombre;

3

+ private int vida;

4

+

5

+ public Soldado(String nombre, int vida) {

6

+ setNombre(nombre);

7

+ setVida(vida);

8

+ }

9

+

10

+ public void setNombre(String nombre) {

11

+ this.nombre = nombre;

12

+ }

13

+

14

+ public void setVida(int vida) {

15

+ this.vida = vida;

16

+ }

17

+

18

+ public String getNombre() {

19

+ return nombre;

20

+ }

21

+

22

+ public int getVida() {

23

+ return vida;

24

+ }

25

+

26

+ public String toString() {

27

+ return "Nombre: " + nombre + ", Vida: " + vida;

28

+ }

29

}

Septimo Commit.

Actualizando Actividad2.java

```
$ code Actividad2.java
```

Octavo Commit / Soldado.java

```
$ git add Actividad2.java
$ git commit -m "Actividad 2"
$ git push
```

Commit

Actividad 2

main

cmestasz committed 12 minutes ago

1 parent e406b59 commit 2b45345

Browse files

Showing 1 changed file with 14 additions and 1 deletion.

Split Unified

15

base01/1a003/Actividad2.java

...

1

// Laboratorio Nro 3 - Actividad 2

2

// Autor: Christian Mestas

3

-

4

public class Actividad2 {

5

public static void main(String[] args) {

6

-

7

}

8

}

15

base01/1a003/Actividad2.java

...

1

// Laboratorio Nro 3 - Actividad 2

2

// Autor: Christian Mestas

3

-

4

+ import java.util.Scanner;

5

+

6

+ public class Actividad2 {

7

+ public static void main(String[] args) {

8

+ Scanner sc = new Scanner(System.in);

9

+ Soldado[] soldados = new Soldado[5];

10

+ for (int i = 0; i < soldados.length; i++) {

11

+ String nombre = sc.nextLine();

12

+ int vida = sc.nextInt();

13

+ sc.nextLine();

14

+ soldados[i] = new Soldado(nombre, vida);

15

+ }

16

+ for (int i = 0; i < soldados.length; i++) {

17

+ System.out.println("Soldado " + (i + 1) + ":");

18

+ System.out.println(soldados[i]);

19

+ }

20

}

21

}

Octavo Commit.

Christian Mestas

Fundamentos de la Programación 2

Página 7

```
$ code Actividad3.java
```

```
$ git add Actividad3.java
$ git commit -m "Actividad 3"
$ git push
```

Noveno Commit.

5. Código desarrollado

Nave.java

```
1 public class Nave {
2     private String nombre;
3     private int fila;
4     private String columna;
5     private boolean estado;
6     private int puntos;
7
8     // Metodos mutadores
9     public void setNombre(String n) {
10         nombre = n;
11     }
12
13     public void setFila(int f) {
14         fila = f;
15     }
16
17     public void setColumna(String c) {
18         columna = c;
19     }
20
21     public void setEstado(boolean e) {
22         estado = e;
23     }
24
25     public void setPuntos(int p) {
26         puntos = p;
27     }
28
29     // Metodos accesoires
30     public String getNombre() {
31         return nombre;
32     }
33
34
35     public int getFila() {
36         return fila;
37     }
38
39
40     public String getColumna() {
41         return columna;
42     }
43
44
45     public boolean getEstado() {
46         return estado;
47     }
48
49
50     public int getPuntos() {
51         return puntos;
52 }
```

```
53     }  
54  
55     public String toString() {  
56         return (nombre + ": (" + fila + ", " + columna + "). Estado: " + estado + ". Puntos:  
57             " + puntos);  
58     }  
59 }
```

- Clase que guarda nombre, fila, columna, estado y puntos de la nave.
- Posee tanto setters como getters para todos los atributos.
- Posee el metodo toString() para poder imprimir el objeto.

DemoBatalla.java

```
1 // Laboratorio Nro 3 - Actividad 1
2 // Autor: Christian Mestas
3 // Colaboro: Marco Aedo, clases DemoBatalla y Nave
4
5 import java.util.*;
6
7 public class DemoBatalla {
8     public static void main(String[] args) {
9         Nave[] misNaves = new Nave[10];
10        Scanner sc = new Scanner(System.in);
11        String nomb, col;
12        int fil, punt;
13        boolean est;
14        for (int i = 0; i < misNaves.length; i++) {
15            System.out.println("Nave " + (i + 1));
16            System.out.print("Nombre: ");
17            nomb = sc.next();
18            System.out.println("Fila ");
19            fil = sc.nextInt();
20            System.out.print("Columna: ");
21            col = sc.next();
22            System.out.print("Estado: ");
23            est = sc.nextBoolean();
24            System.out.print("Puntos: ");
25            punt = sc.nextInt();
26            misNaves[i] = new Nave(); // Se crea un objeto Nave y se asigna su referencia a
                misNaves
27            misNaves[i].setNombre(nomb);
28            misNaves[i].setFila(fil);
29            misNaves[i].setColumna(col);
30            misNaves[i].setEstado(est);
31            misNaves[i].setPuntos(punt);
32            System.out.println();
33        }
34        System.out.println("Naves creadas:");
35        mostrarNaves(misNaves);
36        mostrarPorNombre(misNaves);
37        mostrarPorPuntos(misNaves);
38        System.out.println("Nave con mayor numero de puntos: " +
                mostrarMayorPuntos(misNaves));
39        System.out.println();
40        System.out.println("Naves desordenadas: ");
41        mostrarNaves(desordenar(misNaves));
42    }
43
44    // Metodo para mostrar todas las naves
45    public static void mostrarNaves(Nave[] flota) {
46        for (Nave nave : flota)
47            System.out.println(nave.toString());
48        System.out.println();
49    }
50
51    // Metodo para mostrar todas las naves de un nombre que se pide por teclado
52    public static void mostrarPorNombre(Nave[] flota) {
53        Scanner sc = new Scanner(System.in);
```

```
54     System.out.println("Ingrese el nombre a buscar:");
55     String nombre = sc.nextLine();
56     System.out.println("Naves con el nombre " + nombre + ":");
57     for (Nave nave : flota) {
58         if (nave.getNombre().equals(nombre))
59             System.out.println(nave.toString());
60     }
61     System.out.println();
62 }
63
64 // Metodo para mostrar todas las naves con un numero de puntos inferior o igual
65 // al numero de puntos que se pide por teclado
66 public static void mostrarPorPuntos(Nave[] flota) {
67     Scanner sc = new Scanner(System.in);
68     System.out.println("Ingrese la cantidad de puntos maximos:");
69     int puntos = sc.nextInt();
70     sc.nextLine();
71     System.out.println("Naves con " + puntos + " puntos como maximo:");
72     for (Nave nave : flota) {
73         if (nave.getPuntos() <= puntos)
74             System.out.println(nave.toString());
75     }
76     System.out.println();
77 }
78
79 // Metodo que devuelve la Nave con mayor numero de Puntos
80 public static Nave mostrarMayorPuntos(Nave[] flota) {
81     int maxIdx = 0;
82     for (int i = 0; i < flota.length; i++) {
83         if (flota[i].getPuntos() > flota[maxIdx].getPuntos())
84             maxIdx = i;
85     }
86     return flota[maxIdx];
87 }
88
89 // Crear un metodo que devuelva un nuevo arreglo de objetos con todos los
90 // objetos previamente ingresados
91 // pero aleatoriamente desordenados
92 public static Nave[] desordenar(Nave[] flota) {
93     Nave[] nuevaFlota = new Nave[flota.length];
94     Random r = new Random();
95     System.arraycopy(flota, 0, nuevaFlota, 0, flota.length);
96     for (int idx = 0; idx < nuevaFlota.length; idx++) {
97         int nIdx = r.nextInt(nuevaFlota.length);
98         Nave t = nuevaFlota[idx];
99         nuevaFlota[idx] = nuevaFlota[nIdx];
100        nuevaFlota[nIdx] = t;
101    }
102    return nuevaFlota;
103 }
104 }
```

- Se instancian 10 objetos de la clase nave en el arreglo misNaves, todos los datos son llenados por el usuario.
- mostrarNaves() imprime todas las naves usando su metodo toString().

- `mostrarPorNombre()` imprime todas las naves con el nombre que el usuario ingrese.
- `mostrarPorPuntos()` imprime todas las naves con menor o igual numero de puntos a lo que el usuario ingrese.
- `mostrarMayorPuntos()` retorna la nave que tiene la mayor cantidad de puntos, que luego es impresa.
- `desordenar()` retorna un nuevo arreglo de naves con las naves desordenadas al azar, que luego es impreso con el método `mostrarNaves()`.

Soldado.java

```
1 public class Soldado {
2     private String nombre;
3     private int vida;
4
5     public Soldado(String nombre, int vida) {
6         setNombre(nombre);
7         setVida(vida);
8     }
9
10    public void setNombre(String nombre) {
11        this.nombre = nombre;
12    }
13
14    public void setVida(int vida) {
15        this.vida = vida;
16    }
17
18    public String getNombre() {
19        return nombre;
20    }
21
22    public int getVida() {
23        return vida;
24    }
25
26    public String toString() {
27        return ("Nombre: " + nombre + ". Vida: " + vida);
28    }
29 }
```

- Clase que guarda nombre y vida del soldado.
- Posee tanto setters como getters para los atributos.
- Posee el metodo toString() para poder imprimir el objeto.

Actividad2.java

```
1 // Laboratorio Nro 3 - Actividad 2
2 // Autor: Christian Mestas
3
4 import java.util.Scanner;
5
6 public class Actividad2 {
7     public static void main(String[] args) {
8         Scanner sc = new Scanner(System.in);
9         Soldado[] soldados = new Soldado[5];
10        for (int i = 0; i < soldados.length; i++) {
11            String nombre = sc.nextLine();
12            int vida = sc.nextInt();
13            sc.nextLine();
14            soldados[i] = new Soldado(nombre, vida);
15        }
16        for (int i = 0; i < soldados.length; i++) {
17            System.out.println("Soldado " + (i + 1) + ":");
18            System.out.println(soldados[i]);
19        }
20    }
21 }
```

- Los 5 nombres y sus respectivas vidas son leídas, para luego instanciar objetos de la clase soldado con sus respectivos atributos.
- Luego los atributos son impresos en el orden en el que se ingresaron.

Actividad3.java

```
1 // Laboratorio Nro 3 - Actividad 3
2 // Autor: Christian Mestas
3
4 import java.util.Random;
5
6 public class Actividad3 {
7     public static void main(String[] args) {
8         Soldado[] ejercito1 = inicializarEjercito();
9         Soldado[] ejercito2 = inicializarEjercito();
10        mostrarEjercito(ejercito1, 1);
11        mostrarEjercito(ejercito2, 2);
12        mostrarGanador(ejercito1, ejercito2);
13    }
14
15    public static Soldado[] inicializarEjercito() {
16        Random random = new Random();
17        Soldado[] ejercito = new Soldado[random.nextInt(5) + 1];
18        for (int i = 0; i < ejercito.length; i++)
19            ejercito[i] = new Soldado("Soldado" + i, 0);
20        return ejercito;
21    }
22
23    public static void mostrarEjercito(Soldado[] ejercito, int numero) {
24        System.out.println("Ejercito " + numero + ":");
25        for (int i = 0; i < ejercito.length; i++)
26            System.out.println("Soldado " + (i + 1) + ": " + ejercito[i].getNombre());
27    }
28
29    public static void mostrarGanador(Soldado[] ejercito1, Soldado[] ejercito2) {
30        if (ejercito1.length > ejercito2.length)
31            System.out.println("Gana el ejercito 1!");
32        else if (ejercito1.length < ejercito2.length)
33            System.out.println("Gana el ejercito 2!");
34        else
35            System.out.println("Hubo un empate!");
36    }
37 }
```

- Se crean 2 ejércitos con el método `inicializarEjercito()`, que crea un arreglo de soldados de tamaño al azar entre 1 y 5.
- `mostrarEjercito()` imprime una lista de los soldados de cada ejército.
- `mostrarGanador()` compara los tamaños de los ejércitos y declara al ganador como el ejército mas grande.

6. Ejecución del código

```
Windows PowerShell
Nombre: Jeah
Cita:
Columna: F
Estado: false
Puntos: 6
Nombre: Jeah
Cita:
Columna: C
Estado: false
Puntos: 1
Nombre: Katin
Cita:
Columna: F
Estado: true
Puntos: 14
Nombre: Kian
Cita:
Columna: B
Estado: false
Puntos: 25
Nombre: AAA
Cita:
Columna: J
Estado: false
Puntos: 4
Naves creadas:
Jeah: (5, 2), Estado: true, Puntos: 1
Jeah: (6, 2), Estado: true, Puntos: 12
Jeah: (7, 2), Estado: false, Puntos: 15
Jeah: (8, 2), Estado: false, Puntos: 3
Jeah: (9, 2), Estado: true, Puntos: 6
Jeah: (6, 2), Estado: false, Puntos: 17
Jeah: (6, 2), Estado: false, Puntos: 1
Kian: (9, 2), Estado: true, Puntos: 14
Kian: (9, 2), Estado: false, Puntos: 25
AAA: (6, 2), Estado: false, Puntos: 4
Ingrese el nombre a buscar:
Jeah
Naves con el nombre Jeah:
Jeah: (8, 2), Estado: true, Puntos: 17
Jeah: (3, 2), Estado: false, Puntos: 6
Jeah: (6, 2), Estado: false, Puntos: 1
Ingrese la cantidad de puntos maximos:
10
Naves con 10 puntos como maximo:
Jeah: (7, 2), Estado: true, Puntos: 1
Jeah: (6, 2), Estado: false, Puntos: 3
Jeah: (8, 2), Estado: false, Puntos: 0
Jeah: (8, 2), Estado: false, Puntos: 1
AAA: (6, 2), Estado: false, Puntos: 4
Nave con mayor número de puntos: Kian: (5, 8), Estado: false, Puntos: 25
Naves desordenadas:
Jeah: (7, 8), Estado: false, Puntos: 15
Jeah: (6, 8), Estado: true, Puntos: 12
AAA: (6, 2), Estado: false, Puntos: 3
Kian: (9, 8), Estado: false, Puntos: 25
Jeah: (6, 8), Estado: true, Puntos: 1
Jeah: (8, 8), Estado: false, Puntos: 6
Jeah: (8, 8), Estado: false, Puntos: 1
Jeah: (6, 8), Estado: true, Puntos: 17
Katin: (9, 2), Estado: true, Puntos: 14
```

DemoBatalla.java

```
Windows PowerShell
PS F:\Documents\UNSA\A1S2\FP2 lab\fp2-23b\fase01\lab03> java Actividad2
Jose Luis
56
Perez Manrique
46
Marco Marco
77
Mario Pepe
67
Empirino Josifastico
145
Soldado 1:
Nombre: Jose Luis. Vida: 56
Soldado 2:
Nombre: Perez Manrique. Vida: 46
Soldado 3:
Nombre: Marco Marco. Vida: 77
Soldado 4:
Nombre: Mario Pepe. Vida: 67
Soldado 5:
Nombre: Empirino Josifastico. Vida: 145
PS F:\Documents\UNSA\A1S2\FP2 lab\fp2-23b\fase01\lab03>
```

Actividad2.java

```
Windows PowerShell
PS F:\Documents\UNSA\A1S2\FP2 1ab\fp2-23b\fase01\1ab03> java Actividad3
Ejercito 1:
Soldado 1: Soldado0
Soldado 2: Soldado1
Soldado 3: Soldado2
Soldado 4: Soldado3
Soldado 5: Soldado4
Ejercito 2:
Soldado 1: Soldado0
Soldado 2: Soldado1
Gana el ejercito 1!
PS F:\Documents\UNSA\A1S2\FP2 1ab\fp2-23b\fase01\1ab03>
```

Actividad3.java

7. Estructura de laboratorio 03

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

```
lab03/  
|--- Nave.java  
|--- DemoBatalla.java  
|--- Soldado.java  
|--- Actividad2.java  
|--- Actividad3.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  
|   |--- commit08.jpg  
|   |--- commit09.jpg  
|   |--- ejec01.jpg  
|   |--- ejec02.jpg  
|   |--- ejec03.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 fácil 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 cumple 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 ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Niveles de desempeño

Puntos	Nivel			
	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	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	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		16.5	

9. Referencias

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