



	<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° 04

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programación 2				
TÍTULO DE LA PRÁCTICA:	Arreglos y Objetos Estándar				
NÚMERO DE PRÁCTICA:	04	AÑO LECTIVO:	2024-B	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	12/09/2024	HORA DE PRESENTACIÓN	23:30:05		
INTEGRANTE (s) Subia Huaicane Edson Fabricio				NOTA (0-20)	Nota colocada por el docente
DOCENTE(s): Lino José Pinto Oppe					

RESULTADOS Y PRUEBAS
<p>• EJERCICIOS RESUELTOS:</p> <p>Actividad JUEGO DE NAVES:</p> <ul style="list-style-type: none"> ○ Cree un Proyecto llamado Laboratorio4 ○ Usted podrá reutilizar las dos clases Nave.java y DemoBatalla.java. creadas en Laboratorio 3 ○ Completar el Código de la clase DemoBatalla. <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/Juego_de_Naves.git</p> <p>Mis COMMITS:</p> <ul style="list-style-type: none"> • Este es el primer commit que realicé, en esta versión agregué 2 nuevos métodos: <p>Añadir métodos <code>busquedaLinealNombre</code> y <code>ordenarPorPuntosBurbuja</code>: </p> <p>Para esta nueva versión he añadido dos nuevos métodos.</p> <ul style="list-style-type: none"> - Se implementó el método <code>`busquedaLinealNombre`</code> para buscar una nave por su nombre en el arreglo. - Se añadió el método <code>`ordenarPorPuntosBurbuja`</code> para ordenar el arreglo de naves de menor a mayor por puntos usando el algoritmo de Burbuja.

	<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>

```

DemoBatalla.java
123 + // Metodo para buscar la primera nave con un nombre que se pidio por teclado
124 + public static int busquedaLinealNombre(Nave[] flota, String s) {
125 +     for (int i = 0; i < flota.length; i++) {
126 +         if (flota[i].getNombre().equalsIgnoreCase(s)) {
127 +             return i; // Retorna la posición si encuentra la nave
128 +         }
129 +     }
130 +     return -1; // Retorna -1 si no encuentra la nave
131 + }
132 +
133 + // Método que ordena por número de puntos de menor a mayor (Burbuja)
134 + public static void ordenarPorPuntosBurbuja(Nave[] flota) {
135 +     int n = flota.length;
136 +     for (int i = 0; i < n - 1; i++) {
137 +         for (int j = 0; j < n - i - 1; j++) {
138 +             if (flota[j].getPuntos() > flota[j + 1].getPuntos()) {
139 +                 // Intercambia las naves
140 +                 Nave temp = flota[j];
141 +                 flota[j] = flota[j + 1];
142 +                 flota[j + 1] = temp;
143 +             }
144 +         }
145 +     }
146 + }

```

Aquí destaco la primera versión de los métodos para BúsquedaLineal de las Naves y mostrarlas por PuntosBurbuja según Ordenamiento cada una de ellas.

- **Ésta es la segunda versión de mi juego (demo), donde hemos terminado de crear e implementar los métodos necesarios:**

feat: Añadir métodos para ordenación por nombre con Burbuja y búsqueda b...

_inaria por nombre

- Se implementó el método `ordenarPorNombreBurbuja` para ordenar el arreglo de naves de A a Z por nombre.
- Se añadió el método `busquedaBinariaNombre` para buscar una nave por su nombre utilizando el algoritmo de búsqueda binaria.

```

147 + // Método que ordena por nombre de A a Z utilizando el algoritmo de Burbuja
148 + public static void ordenarPorNombreBurbuja(Nave[] flota) {
149 +     int n = flota.length;
150 +     for (int i = 0; i < n - 1; i++) {
151 +         for (int j = 0; j < n - i - 1; j++) {
152 +             if (flota[j].getNombre().compareToIgnoreCase(flota[j + 1].getNombre()) > 0) {
153 +                 // Intercambio de naves
154 +                 Nave temp = flota[j];
155 +                 flota[j] = flota[j + 1];
156 +                 flota[j + 1] = temp;
157 +             }
158 +         }
159 +     }
160 + }

```

```

161 + // Método para buscar una nave por su nombre utilizando búsqueda binaria (requiere que esté ordenado por nombre)
162 + public static int busquedaBinariaNombre(Nave[] flota, String s) {
163 +     int inicio = 0;
164 +     int fin = flota.length - 1;
165 +
166 +     while (inicio <= fin) {
167 +         int medio = (inicio + fin) / 2;
168 +         int comparacion = flota[medio].getNombre().compareToIgnoreCase(s);
169 +
170 +         if (comparacion == 0) {
171 +             return medio; // Retorna la posición si encuentra el nombre
172 +         } else if (comparacion < 0) {
173 +             inicio = medio + 1;
174 +         } else {
175 +             fin = medio - 1;
176 +         }
177 +     }
178
179 +     return -1; // Retorna -1 si no encuentra ninguna nave con ese nombre
180 + }

```

- **Para esta versión agregué los métodos de Ordenamiento por Selección:**

Añadir métodos para ordenación por puntos y por nombre con Selección. 📄



Para esta versión:

- Se implementó el método `ordenarPorPuntosSeleccion` para ordenar el arreglo de naves de menor a mayor por puntos.
- Se añadió el método `ordenarPorNombreSeleccion` para ordenar el arreglo de naves de A a Z por nombre utilizando el algoritmo de Selección.

```

181 + // Método que ordena por número de puntos de menor a mayor utilizando el algoritmo de Selección
182 + public static void ordenarPorPuntosSeleccion(Nave[] flota) {
183 +     int n = flota.length;
184 +     for (int i = 0; i < n - 1; i++) {
185 +         int minIndex = i;
186 +         for (int j = i + 1; j < n; j++) {
187 +             if (flota[j].getPuntos() < flota[minIndex].getPuntos()) {
188 +                 minIndex = j;
189 +             }
190 +         }
191 +         // Intercambio de naves
192 +         Nave temp = flota[minIndex];
193 +         flota[minIndex] = flota[i];
194 +         flota[i] = temp;
195 +     }
196 + }

```

	<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: 4</p>

```

197 +
198 + // Método que ordena por nombre de A a Z utilizando el algoritmo de Selección
199 + public static void ordenarPorNombreSeleccion(Nave[] flota) {
200 +     int n = flota.length;
201 +     for (int i = 0; i < n - 1; i++) {
202 +         int minIndex = i;
203 +         for (int j = i + 1; j < n; j++) {
204 +             if (flota[j].getNombre().compareToIgnoreCase(flota[minIndex].getNombre()) < 0) {
205 +                 minIndex = j;
206 +             }
207 +         }
208 +         // Intercambio de naves
209 +         Nave temp = flota[minIndex];
210 +         flota[minIndex] = flota[i];
211 +         flota[i] = temp;
212 +     }
213 + }

```

- **En este commit, agregué los métodos para ordenamiento por Inserción:**

Añadir métodos para ordenación por puntos y por nombre con Inserción



Lo que se realizó en la siguiente versión fue:

- Se implementó el método `ordenarPorPuntosInsercion` para ordenar el arreglo de naves de menor a mayor por puntos utilizando el algoritmo de Inserción.
- Se añadió el método `ordenarPorNombreInsercion` para ordenar el arreglo de naves de Z a A por nombre utilizando el algoritmo de Inserción.

```

214 + // Método que ordena por número de puntos de menor a mayor utilizando el algoritmo de Inserción
215 + public static void ordenarPorPuntosInsercion(Nave[] flota) {
216 +     int n = flota.length;
217 +     for (int i = 1; i < n; i++) {
218 +         Nave clave = flota[i];
219 +         int j = i - 1;
220 +         while (j >= 0 && flota[j].getPuntos() > clave.getPuntos()) {
221 +             flota[j + 1] = flota[j];
222 +             j--;
223 +         }
224 +         flota[j + 1] = clave;
225 +     }
226 + }
227 +

```

	<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: 5</p>

```

228 + // Método que ordena por nombre de Z a A utilizando el algoritmo de Inserción
229 + public static void ordenarPorNombreInsercion(Nave[] flota) {
230 +     int n = flota.length;
231 +     for (int i = 1; i < n; i++) {
232 +         Nave clave = flota[i];
233 +         int j = i - 1;
234 +         while (j >= 0 && flota[j].getNombre().compareToIgnoreCase(clave.getNombre()) < 0) {
235 +             flota[j + 1] = flota[j];
236 +             j--;
237 +         }
238 +         flota[j + 1] = clave;
239 +     }
240 + }

```

- **Para la versión final agregué los llamados a los nuevos métodos en el Main y los enumeré para seguir la lógica correspondiente**

Versión final de esta versión 5.3.1: ↕



Lo realizado para esta versión fue:

Se actualizó el método main para incluir llamadas a los nuevos métodos y gestionar la lógica de presentación, así como también se enumeraron los comentarios que hacen mención a los métodos o funciones más importantes del programa.

```

51 + // 8. Ordenar naves por nombre
52 + ordenarPorNombreBurbuja(misNaves);
53 + System.out.println("\nNaves ordenadas por nombre (Burbuja:");
54 + mostrarNaves(misNaves);
55 +
56 + // 9. Buscar una nave por nombre
57 + System.out.print("Ingrese el nombre de la nave a buscar: ");
58 + String nombreBuscado = scanPro.next();
59 + int indice = busquedaBinariaNombre(misNaves, nombreBuscado);
60 + if (indice != -1) {
61 +     System.out.println("Nave encontrada: " + misNaves[indice]);
62 + } else {
63 +     System.out.println("No se encontró la nave con el nombre: " + nombreBuscado);
64 + }
65 +
66 + // 10. Ordenar naves por puntos
67 + ordenarPorPuntosSeleccion(misNaves);
68 + System.out.println("\nNaves ordenadas por puntos (Selección:");
69 + mostrarNaves(misNaves);
70 +
71 + // 11. Ordenar naves por nombre (Selección)
72 + ordenarPorNombreSeleccion(misNaves);
73 + System.out.println("\nNaves ordenadas por nombre (Selección:");
74 + mostrarNaves(misNaves);

```

	<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>

En la siguiente sección mostraré la ejecución de la versión final de mi código fuente del programa, trabajado en Visual Studio, en cada captura de pantalla se visualizará el buen funcionamiento de los nuevos métodos adicionados y fundamentados correctamente. (El código fuente se visualiza mucho mejor en mi repositorio)

```
Nave 1
Nombre: Robert
Fila: 3
Columna: 4
Estado (true para operativa, false para inactiva): True
Puntos: 10
Nave 2
Nombre: Loco
Fila: 4
Columna: 4
Estado (true para operativa, false para inactiva): True
Puntos: 9
Nave 3
Nombre: BROH
Fila: 6
Columna: 5
Estado (true para operativa, false para inactiva): True
Puntos: 9
```

```
Naves creadas:
Nave [Nombre: Robert, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]
Nave [Nombre: Loco, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]
Nave [Nombre: BROH, Fila: 6, Columna: 5, Estado: Operativa, Puntos: 9]
Ingrese el nombre de la nave a mostrar:
Robert
Nave [Nombre: Robert, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]
Ingrese el límite de puntos: 8
No se encontraron naves con puntos menores o iguales a: 8

Nave con mayor número de puntos: Nave [Nombre: Robert, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]
```

```
Naves desordenadas aleatoriamente:
Nave [Nombre: Robert, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]
Nave [Nombre: Loco, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]
Nave [Nombre: BROH, Fila: 6, Columna: 5, Estado: Operativa, Puntos: 9]

Naves ordenadas por nombre (Burbuja):
Nave [Nombre: BROH, Fila: 6, Columna: 5, Estado: Operativa, Puntos: 9]
Nave [Nombre: Loco, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]
Nave [Nombre: Robert, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]
Ingrese el nombre de la nave a buscar: Loco
Nave encontrada: Nave [Nombre: Loco, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]



Naves ordenadas por puntos (Selección):
Nave [Nombre: BROH, Fila: 6, Columna: 5, Estado: Operativa, Puntos: 9]
Nave [Nombre: Loco, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]
Nave [Nombre: Robert, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]

Naves ordenadas por nombre (Selección):
Nave [Nombre: BROH, Fila: 6, Columna: 5, Estado: Operativa, Puntos: 9]
Nave [Nombre: Loco, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]
Nave [Nombre: Robert, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]

Naves ordenadas por puntos (Inserción):
Nave [Nombre: BROH, Fila: 6, Columna: 5, Estado: Operativa, Puntos: 9]
Nave [Nombre: Loco, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]
Nave [Nombre: Robert, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]

Naves ordenadas por nombre (Inserción):
Nave [Nombre: BROH, Fila: 6, Columna: 5, Estado: Operativa, Puntos: 9]
Nave [Nombre: Loco, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]
Nave [Nombre: Robert, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]
```

EJECUCIÓN DEL PROGRAMA (v5.3.0): “JUEGO DE NAVES”

	<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>

• PRUEBAS

¿Con qué valores comprobaste que tu práctica estuviera correcta?

Comprobé la práctica utilizando diversas combinaciones de entradas al crear las naves en el juego. Ingresé nombres de naves como "Loco", "Robert", y "X-Falcon", variando los valores para la fila (por ejemplo, 1, 2, 3), columna (A, B, C), estado (true y false), y puntos (10, 20, 30). Para probar los nuevos métodos, utilicé nombres existentes y no existentes al buscar naves por nombre, así como diferentes límites de puntos para las búsquedas de naves según sus puntajes. Además, utilicé métodos de ordenamiento como burbuja, selección e inserción en naves con diversos valores de puntos para evaluar su eficacia.

¿Qué resultado esperabas obtener para cada valor de entrada?



Esperaba que el juego tuviera el siguiente comportamiento:

- Al ordenar naves por puntos (método de burbuja, selección e inserción): Se esperaba que las naves se organizaran correctamente de acuerdo a sus puntos, reflejando el orden ascendente esperado después de aplicar cada método.*
- Al buscar naves por nombre (método de búsqueda lineal): El programa encontraría correctamente las naves que coincidieran con el nombre ingresado y, si no existían, mostraría un mensaje informativo.*
- Al buscar naves por nombre (método de búsqueda binaria): Este método debía identificar las naves de forma rápida y eficiente después de haber sido ordenadas, mostrando los resultados esperados o un mensaje de no coincidencia.*
- Al mostrar la nave con mayor cantidad de puntos: Se esperaba que se identificara correctamente la nave con el mayor puntaje en el arreglo.*
- Al desordenar naves: Se generaría un nuevo arreglo con las naves en un orden aleatorio.*

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

Los resultados obtenidos fueron los esperados:

- Al ordenar naves por puntos: Las naves se mostraron correctamente ordenadas en función de sus puntajes, validando así la funcionalidad de los métodos de ordenamiento (burbuja, selección, inserción) utilizados.*
- Al buscar naves por nombre (búsqueda lineal): El programa encontró las naves correctamente cuando se ingresaron nombres válidos y mostró un mensaje cuando no se encontró coincidencia.*
- Al buscar naves por nombre (búsqueda binaria): El programa identificó correctamente las naves después de haber sido ordenadas, demostrando la eficacia de este método en comparación con la búsqueda lineal.*
- Al mostrar la nave con mayor cantidad de puntos: El programa identificó correctamente la nave con el mayor puntaje en el arreglo.*
- Al desordenar naves: Se creó un nuevo arreglo con las naves en un orden aleatorio, validando así la funcionalidad del método de desordenamiento.*

	<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: 8</p>

1. CUESTIONARIO:

Los datos más importantes que consideré para realizar el código del juego de Naves, especialmente en relación con los nuevos métodos de ordenamiento, fueron los siguientes:



1. *Nave: Cada nave tiene atributos como nombre, fila, columna, estado y puntos. Estos atributos son esenciales para definir las características y el estado de cada nave en el juego, y son los criterios clave para los métodos de ordenamiento.*
2. *Arreglo de naves: El uso de un arreglo para almacenar múltiples objetos de tipo Nave es fundamental. Este arreglo permite gestionar de manera eficiente varias naves durante la partida y facilita la aplicación de algoritmos de ordenamiento, como burbuja, selección e inserción.*
3. *Métodos de ordenamiento: La implementación de nuevos métodos de ordenamiento es crucial para organizar las naves según diferentes criterios, como sus puntos. Por ejemplo:*
 - *Ordenamiento por burbuja: Permite organizar las naves de menor a mayor puntaje, lo que es útil para determinar cuáles naves tienen menor rendimiento en el juego.*
 - *Ordenamiento por selección: Este método también organiza las naves, pero con un enfoque diferente, lo que permite comparar la eficiencia de diferentes algoritmos de ordenamiento.*
 - *Ordenamiento por inserción: Este método es efectivo para ordenar las naves en tiempo real, lo que puede ser útil si se añaden nuevas naves dinámicamente durante la ejecución del juego.*
4. *Interacción con el usuario: Los métodos de ordenamiento mejoran la interacción con el usuario al permitirle ver las naves organizadas según sus criterios preferidos. Esto agrega una dimensión de estrategia y análisis al juego, ya que los jugadores pueden evaluar fácilmente el rendimiento de sus naves.*
5. *Comparación y actualización: Los métodos de comparación utilizados en el ordenamiento son fundamentales para aplicar la lógica del juego. Permiten determinar cómo se deben organizar las naves en función de sus puntajes, y también pueden ser utilizados para identificar las naves que superan ciertos umbrales de puntos.*

CONCLUSIONES

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

La implementación del juego de Naves ha destacado por su estructura clara y la interacción fluida que proporciona al usuario. La incorporación de métodos de ordenamiento para gestionar las naves según sus atributos, como los puntos, ha mejorado significativamente la experiencia del jugador, facilitando la visualización y comparación del rendimiento de cada nave.

Los resultados obtenidos confirmaron que estos métodos respondieron eficazmente a las entradas del usuario, permitiendo una gestión más dinámica de la flota. Este proyecto no solo aplicó conceptos fundamentales de programación orientada a objetos, sino que también demostró la importancia del ordenamiento en la lógica del juego, enriqueciendo así el aprendizaje sobre diseño de juegos.

	<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>

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

Colocare 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/FundProq2/blob/main/entregaLaboratorio01/Hilacondo Emanuel LABORATORIO_01.pdf](https://github.com/LINOPINTO2023/FundProq2/blob/main/entregaLaboratorio01/Hilacondo_Emanuel_LABORATORIO_01.pdf)

https://github.com/Q3son/Juego_de_Naves.git

	<p 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: 10

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	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	3	
TOTAL		20	8	18	