

# Informe de Laboratorio 12

## Tema: Definición de Clases de Usuario

Nota

Estudiante	Escuela	Asignatura
Jhonatan David Arias Quispe jariasq@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de Programación 2 Semestre: II Código: 1701213

Laboratorio	Tema	Duración
12	Definición de Clases de Usuario	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 4 Diciembre 2023	Al 11 Diciembre 2023

### 1. Actividades

- Al ejecutar el videojuego, el programa deberá dar las opciones:
- 1. Juego rápido (tal cual como en el laboratorio 11) Al acabar el juego mostrar las opciones de volver a jugar y de volver al menú principal. También se deberá tener la posibilidad de cancelar el juego actual en cualquier momento, permitiendo escoger entre empezar un juego totalmente nuevo o salir al menú principal.
- 2. Juego personalizado: permite gestionar ejércitos. Primero se generan los 2 ejércitos con sus respectivos soldados y se muestran sus datos. Luego se tendrá que escoger cuál de los 2 ejércitos se va a gestionar, después se mostrarán las siguientes opciones:
  - Crear Soldado: permitirá crear un nuevo soldado personalizado y añadir al final del ejército (recordar que límite es de 10 soldados por ejército)
  - Eliminar Soldado (no debe permitir un ejército vacío)
  - Clonar Soldado (crea una copia exacta del soldado) y se añade al final del ejército (recordar que límite es de 10 soldados por ejército)
  - Modificar Soldado (con submenú para cambiar alguno de los atributos nivelAtaque, nivelDefensa, vidaActual)
  - Comparar Soldados (verifica si atributos: nombre, nivelAtaque, nivelDefensa, vidaActual y vive son iguales)
  - Intercambiar Soldados (intercambia 2 soldados en sus posiciones en la estructura de datos del ejército)

- Ver soldado (Búsqueda por nombre)
  - Ver ejército
  - Sumar niveles (usando Method-Call Chaining), calcular las sumatorias de nivelVida, nivelAtaque, nivelDefensa, velocidad de todos los soldados de un ejército
    - Por ejemplo, si ejército tendría 3 soldados:
    - `s=s1.sumar(s2).sumar(s3);`
    - `s` es un objeto Soldado nuevo que contendría las sumatorias de los 4 atributos indicados de los 3 soldados. Ningún soldado cambia sus valores
  - Jugar (se empezará el juego con los cambios realizados) y con las mismas opciones de la opción 1.
  - Volver (muestra el menú principal) Después de escoger alguna de las opciones 1) a 9) se podrá volver a elegir uno de los ejércitos y se mostrarán las opciones 1) a 11)
- 3. Salir

## 2. Equipos, materiales y temas utilizados

- Sistema Operativo ArchCraft GNU Linux 64 bits Kernell
- NeoVim
- OpenJDK 64-Bit 20.0.1
- Git 2.42.0
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.
- Creacion de programas con CLI
- Biblioteca Graphics (origen propio)

## 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/JhonatanDczel/fp2-23b.git>
- URL para el laboratorio 12 en el Repositorio GitHub.
- <https://github.com/JhonatanDczel/fp2-23b/tree/main/fase02/lab12>
- El trabajo de este laboratorio es en su mayor parte lo mismo que en otros laboratorios, por lo que las variaciones serán mínimas

## 4. Proyecto lab12

- Creamos un directorio en fase02 que contenga los archivos del laboratorio y copiamos los archivos del anterior laboratorio.
- Para el tablero se usará un array bidimensional simple.
- La estructura del laboratorio presente es:

```
>>> lab12 git:(main) tree
.
├── latex
│   ├── acts
│   │   ├── a1.tex
│   │   ├── a2.tex
│   │   ├── a3.tex
│   │   ├── a4.tex
│   │   ├── a5.tex
│   │   ├── commits.tex
│   │   ├── exec.tex
│   │   └── main.tex
│   ├── foot
│   │   ├── rubricas.aux
│   │   └── rubricas.tex
│   ├── head
│   │   ├── codeFormat.tex
│   │   ├── dataTable.tex
│   │   ├── format.tex
│   │   ├── labData.tex
│   │   └── pkgs.tex
│   └── img
│       ├── exec.png
│       ├── logo_abet.png
│       ├── logo_episunsa.png
│       ├── logo_unsa.jpg
│       └── tree.jpg
├── informe-latex.aux
├── informe-latex.log
├── informe-latex.out
├── informe-latex.pdf
├── informe-latex.tex
├── Soldado.class
├── Soldado.java
├── VideoJuego.class
└── VideoJuego.java

6 directories, 29 files
```

## 5. Clase Soldado

- Los atributos para la clase soldado son:

```
1 private String nombre;
2 private int fila;
3 private int nivelAtaque = random(5);
4 private int nivelDefensa = random(5);
5 private int columna;
6 private int nivelVida;
7 private int vidaActual;
8 private int velocidad;
9 private String actitud;
10 private boolean vive;
11 private String team;
```

- Cada atributo que lo requiere, tiene sus metodos setters y getters para encapsular la información.
- Se usan 3 constructores sobrecargados que son los siguientes:

```
1 public Soldado(String t) {
2     team = t;
3     velocidad = 0;
4     vive = true;
5     actitud = "ataque";
6
7 }
8 public Soldado(int v, String t) {
9     team = t;
10    velocidad = v;
11    vive = true;
12    actitud = "ataque";
13 }
14 public Soldado(int v, int nV, String t) {
15     team = t;
16     vive = true;
17     velocidad = v;
18     nivelVida = nV;
19     actitud = "ataque";
20 }
```

- Estos constructores nos servirán cuando vayamos a crear soldados con distintos datos base.
- Adicionalmente tenemos los metodos de accion del soldado:

```
1 public void atacar() {
2     actitud = "ofensiva";
3 }
4 public void defender() {
5     actitud = "defensiva";
6 }
7 public void huir() {
8     actitud = "fuga";
9     velocidad += 2;
10 }
11 public void avanzar() {
12     velocidad += 1;
13 }
14
15 public void serAtacado() {
16     vidaActual -= 1;
17     if(vidaActual == 0) morir();
18 }
19 public void morir() {
20     vive = false;
21 }
22
23 public void retroceder() {
24     if (velocidad > 0) {
25         velocidad = 0;
26         actitud = "defensiva";
27     } else if (velocidad == 0) {
28         velocidad = -1;
29     }
30 }
```

- Las acciones cambian los estados de los atributos del soldado, a los que accederemos despues con los metodos accesores:

```
1 public String getTeam() {
2     return team;
3 }
4
5 public void setNombre(String n) {
6     nombre = n;
7 }
8
9 public void setFila(int f) {
10    fila = f;
11 }
12
13 public void setColumna(int c) {
14    columna = c;
15 }
16
17 public void setNivelVida(int p) {
18    nivelVida = p;
19 }
20
21 public String getNombre() {
22    return nombre;
23 }
24
25 public int getFila() {
26    return fila;
27 }
28
29 public int getColumna() {
30    return columna;
31 }
32
33 public int getNivelVida() {
34    return nivelVida;
35 }
36
37 public int getNivelAtaque() {
38    return nivelAtaque;
39 }
40
41 public int getNivelDefensa() {
42    return nivelDefensa;
43 }
44
45 public void setNivelAtaque(int n) {
46    nivelAtaque = n;
47 }
48
49 public void setNivelDefensa(int n) {
50    nivelDefensa = n;
51 }
52
53 public boolean isLive() {
54    return vive;
55 }
```

- Estos metodos accesoros nos servirán para manejar la lógica interna del videojuego
- Adicionalmente tenemos algunos metodos auxiliares que usamos en la misma clase:

```
1 public String toString() {
2     return "Nombre: " + nombre +
```

```
3      " | Ubicacion: " + fila + ", " + column +  
4      " | nivelVida: " + nivelVida +  
5      " | Estado: " + (vive ? "Vivo" : "Muerto") +  
6      " | Actitud: "+ actitud +"\n" ;  
7  }  
8  private int random(int n) {  
9      return (int) (Math.random() * n + 1);  
10 }
```

## 6. Clase VideoJuego

La clase VideoJuego contiene la logica principal y es la clase que contiene el metodo main, por lo tanto, el que debería llamarse para ejecutar el juego. La estructura de la clase esta dada de la siguiente manera:

- Metodos:
- juegoRapido
- juegoPersonalizado
- crearSoldado
- eliminarSoldado
- clonarSoldado
- modificarSoldado
- compararSoldado
- intercambiarSoldado
- verSoldado
- verEjercito
- sumarNiveles
- jugar
- volver
- main
- fillTable
- addSoldado
- printTable
- printArr
- random
- printMayorNivel
- printPromedioPuntos
- printPuntosAll

- printSoladosOrdenados
- printRankingPointsBubble
- printRankingPointsSelect
- getWinner
- randomWinner
- mover
- atacar

A continuación veremos la explicación de los métodos principales para el funcionamiento.

## 7. Juego Rapido

El método `juegoRapido()` representa la implementación de un juego estratégico entre dos ejércitos de soldados dispuestos en una matriz bidimensional. A continuación, se proporciona una explicación detallada del código:

### 7.1. Configuración Inicial

La función comienza solicitando al usuario ingresar 'S' si desea salir de la partida. Si se ingresa 'S', la función termina inmediatamente. En caso contrario, se procede a la configuración inicial del juego.

```
1 Scanner sc = new Scanner(System.in);
2 System.out.println("Para salir de la partida ingresa 'S'");
3 String salir = sc.next();
4 if (salir.toLowerCase().charAt(0) == 's') return;
```

### 7.2. Inicialización de Variables

Se crea una matriz `table` de dimensiones 10x10 para representar el campo de batalla, y dos listas (`ejercito1` y `ejercito2`) que contendrán instancias de la clase `Soldado`. La función `fillTable()` se llama para inicializar la matriz y las listas.

```
1 Soldado[][] table = new Soldado[10][10];
2 ArrayList<Soldado> ejercito1 = new ArrayList<Soldado>();
3 ArrayList<Soldado> ejercito2 = new ArrayList<Soldado>();
4 fillTable(table, ejercito1, ejercito2);
```

### 7.3. Impresión de Información Inicial

Se imprime en consola el soldado con mayor nivel en cada ejército y, posteriormente, se muestra el promedio de puntos de ambos ejércitos.

```
1 printMayorNivel(table, ejercito1, ejercito2);
2 System.out.println("-----");
3 System.out.println("IMPRIMIR PROMEDIO DE PUNTOS");
4 printPromedioPuntos(table, ejercito1, ejercito2);
5 System.out.println("-----");
```

## 7.4. Ordenamiento y Ranking

Se imprime la lista de soldados ordenada por algún criterio no especificado y se muestra el ranking de puntos de ambos ejércitos, utilizando algoritmos de ordenamiento Bubble Sort y Select Sort, respectivamente.

```
1 System.out.println("-----");
2 System.out.println("SOLDADOS ORDENADOS");
3 printSoldadosOrdenados(table, ejercito1, ejercito2);
4 System.out.println("-----");
5 System.out.println("RANKING DE PUNTOS EJERCITO1 POR BUBBLE SORT");
6 printRankingPointsBubble(table, ejercito1);
7 System.out.println("-----");
8 System.out.println("RANKING DE PUNTOS EJERCITO2 POR SELECT SORT");
9 printRankingPointsSelect(table, ejercito2);
```

## 7.5. Desarrollo del Juego

Se inicia un bucle que representa los turnos del juego. En cada turno, se muestra el estado actual del campo de batalla (`printTable(table)`) y se realiza un movimiento de los soldados. Se alterna entre los dos ejércitos en cada turno.

```
1 int turno = 1;
2 while (true) {
3     printTable(table);
4     mover(table, ejercito1, ejercito2, turno);
5
6     turno = turno == 1 ? 2 : 1;
7
8     String winner = getWinner(ejercito1, ejercito2);
9
10    if (winner != null) {
11        System.out.println("#####");
12        System.out.println("GANA EL EQUIPO " + winner);
13        System.out.println("#####");
14        break;
15    }
16 }
```

## 7.6. Conclusión del Juego

Cuando se determina un ganador (mediante la función `getWinner()`), se imprime el equipo ganador y se finaliza el juego.

Entonces, el método `juegoRapido()` encapsula la lógica de un juego estratégico entre dos ejércitos de soldados, proporcionando información inicial, estadísticas y una interfaz para que los usuarios participen en turnos alternativos hasta que se determine un ganador.

## 8. Juego Personalizado

El método `juegoPersonalizado` proporciona una interfaz interactiva para que el usuario personalice y gestione los soldados de dos equipos en un juego de estrategia. Cada opción del menú está asociada a una función específica que realiza acciones como crear, eliminar, clonar o modificar soldados, así como visualizar información sobre los mismos y el estado general de los ejércitos. el método también incluye opciones para avanzar al juego o volver al menú principal. A continuación se detalla su estructura interna:



## 8.1. Inicialización de Variables

Se utiliza un objeto `Scanner` para la entrada de datos desde la consola. Se crea una matriz llamada `table` de dimensiones  $10 \times 10$  para representar el campo de juego. Se declaran dos `ArrayList` llamados `ejercito1` y `ejercito2` para almacenar los soldados de cada equipo.

```
1 Scanner sc = new Scanner(System.in);
2 Soldado[][] table = new Soldado[10][10];
3 ArrayList<Soldado> ejercito1 = new ArrayList<Soldado>();
4 ArrayList<Soldado> ejercito2 = new ArrayList<Soldado>();
5 fillTable(table, ejercito1, ejercito2);
```

## 8.2. Rellenar la Matriz y Listas de Soldados

Se invoca el método `fillTable` para inicializar la matriz y las listas de soldados de ambos equipos.

```
1 fillTable(table, ejercito1, ejercito2);
```

## 8.3. Impresión de Soldados Ordenados

Se imprime en la consola la información de los soldados ordenados en el campo de juego.

```
1 System.out.println("SOLDADOS ORDENADOS");
2 printSoldadosOrdenados(table, ejercito1, ejercito2);
3 System.out.println("#####");
```

## 8.4. Selección del Equipo a Personalizar

Se solicita al usuario que ingrese el símbolo del equipo ("\*" o "#") que desea personalizar. Se determina el `ArrayList` del equipo seleccionado mediante un operador ternario.

```
1 System.out.print("Escoge al equipo a personalizar: ");
2 String team = sc.next();
3 ArrayList<Soldado> currentE = team.equals("*") ? ejercito1 : ejercito2;
```

## 8.5. Menú de Opciones

Se presenta un menú con varias opciones numeradas que permiten al usuario realizar acciones específicas en relación con los soldados y equipos.

```
1 System.out.println("Escoge una opcion: ");
2 // ... (Listado de opciones)
3 int input = sc.nextInt();
```

## 8.6. Ejecución de Acciones Según la Opción Seleccionada

Se utiliza una serie de condicionales para determinar la acción a realizar según la opción ingresada por el usuario.

```
1 if (input == 1) crearSoldado(table, currentE, team);
2 if (input == 2) eliminarSoldado(table, currentE, team);
3 // ... (Resto de opciones)
4 if (input == 10) jugar();
```

```
5 if (input == 11) volver();
```

## 9. Métodos de gestión de Soldados

Estas funciones proporcionan al usuario la capacidad de crear, eliminar, clonar y modificar soldados en su ejército, contribuyendo así a la personalización y estrategia dentro del juego. Cada función interactúa directamente con la matriz que representa el campo de juego y la lista de soldados del equipo correspondiente.

### 9.1. Función crearSoldado

**Propósito:** Permite al usuario crear un nuevo soldado para su ejército.

**Parámetros:**

- `Soldado[] [] t`: La matriz que representa el campo de juego.
- `ArrayList<Soldado>e`: Lista de soldados del equipo actual.
- `String team`: Símbolo del equipo actual.

**Descripción:**

```
1 public static void crearSoldado(Soldado[] [] t, ArrayList<Soldado> e, String team) {
2     Scanner sc = new Scanner(System.in);
3     Soldado s = new Soldado(team);
4
5     if(e.size() >= 10) {
6         System.out.println("El ejercito tiene el maximo de soldados: ");
7         return;
8     }
9
10    System.out.print("Ingresa el nivel del soldado: ");
11    s.setNivelVida(sc.nextInt());
12    System.out.print("Ingresa el nivel de ataque: ");
13    s.setNivelAtaque(sc.nextInt());
14    System.out.print("Ingresa el nivel de defensa: ");
15    s.setNivelDefensa(sc.nextInt());
16    System.out.print("Ingresa la columna: ");
17    s.setColumna(sc.nextInt());
18    System.out.print("Ingresa la fila: ");
19    s.setFila(sc.nextInt());
20    s.setNombre("Soldado " + (e.size() + 1));
21
22    e.add(s);
23    t[s.getColumna()][s.getFila()] = s;
24
25 }
```

### 9.2. Función eliminarSoldado

**Propósito:** Permite al usuario eliminar un soldado de su ejército.

**Parámetros:**

- `Soldado[] [] t`: La matriz que representa el campo de juego.
- `ArrayList<Soldado>e`: Lista de soldados del equipo actual.
- `String team`: Símbolo del equipo actual.

**Descripción:**

```
1 public static void eliminarSoldado(Soldado[][] t, ArrayList<Soldado> e, String team) {
2     Scanner sc = new Scanner(System.in);
3
4     if(e.size() <= 1) {
5         System.out.println("El ejercito nesecita al menos 1 soldado");
6         return;
7     }
8
9     System.out.print("Ingresa la columna: ");
10    int col = sc.nextInt();
11    System.out.print("Ingresa la fila: ");
12    int fila = sc.nextInt();
13
14    for(int i = 0; i < e.size(); i += 1) {
15        if(e.get(i).getColumna() == col && e.get(i).getFila() == fila){
16            e.remove(i);
17            break;
18        }
19    }
20
21    t[col][fila] = null;
22 }
23 }
```

**9.3. Función clonarSoldado**

**Propósito:** Permite al usuario clonar un soldado existente en una nueva posición.

**Parámetros:**

- Soldado[][] t: La matriz que representa el campo de juego.
- ArrayList<Soldado>e: Lista de soldados del equipo actual.
- String team: Símbolo del equipo actual.

**Descripción:**

```
1 public static void clonarSoldado(Soldado[][] t, ArrayList<Soldado> e, String team) {
2     Scanner sc = new Scanner(System.in);
3
4     System.out.print("Ingresa la columna: ");
5     int col = sc.nextInt();
6     System.out.print("Ingresa la fila: ");
7     int fila = sc.nextInt();
8
9     Soldado soldadoOriginal = t[col][fila];
10
11     Soldado soldadoNuevo = new Soldado(soldadoOriginal.getTeam());
12     soldadoNuevo.setNivelAtaque(soldadoOriginal.getNivelAtaque());
13     soldadoNuevo.setNivelDefensa(soldadoOriginal.getNivelDefensa());
14     soldadoNuevo.setNivelVida(soldadoOriginal.getNivelVida());
15     soldadoNuevo.setNombre("Soldado " + (e.size() + 1));
16
17     System.out.print("Ingresa la columna nueva: ");
18     int newCol = sc.nextInt();
19     System.out.print("Ingresa la fila nueva: ");
20     int newFila = sc.nextInt();
21
22     e.add(soldadoNuevo);
23     t[newCol][newFila] = soldadoNuevo;
24 }
```

## 9.4. Función modificarSoldado

**Propósito:** Permite al usuario modificar las características y posición de un soldado existente.

**Parámetros:**

- Soldado[] [] t: La matriz que representa el campo de juego.
- ArrayList<Soldado>e: Lista de soldados del equipo actual.
- String team: Símbolo del equipo actual.

**Descripción:**

```
1 public static void modificarSoldado(Soldado[] [] t, ArrayList<Soldado> e, String team) {
2     Scanner sc = new Scanner(System.in);
3
4     System.out.print("Ingresa la columna: ");
5     int col = sc.nextInt();
6     System.out.print("Ingresa la fila: ");
7     int fila = sc.nextInt();
8
9     Soldado s = t[col][fila];
10    System.out.println("Ingresa nivel de vida nuevo: ");
11    s.setNivelVida(sc.nextInt());
12    System.out.println("Ingresa nivel de ataque nuevo: ");
13    s.setNivelAtaque(sc.nextInt());
14    System.out.println("Ingresa nivel de defensa nuevo: ");
15    s.setNivelDefensa(sc.nextInt());
16    System.out.println("Ingresa la columna nueva: ");
17    s.setColumna(sc.nextInt());
18    System.out.println("Ingresa la fila nueva: ");
19    s.setFila(sc.nextInt());
20 }
```

## 10. Ejecucion del juego

**Selección de la Opción "Juego rapido":**

El usuario elige la opción "Juego rapido" ingresando el número 1 en el menú principal.

```
1 1. Juego rapido
2 2. Juego personalizado
3 3. Salir
4 1
```

**Inicio del Juego Rapido:**

Se presenta un mensaje indicando que para salir de la partida, el usuario debe ingresar 'S'.

```
1 Para salir de la partida ingresa 'S'
```

**Información de los Soldados con Mayor Nivel de Puntos:**

Se muestran los soldados con el mayor nivel de puntos para la primera y segunda flota.

```
1 Los soldados con el mayor nivel de puntos de la primera flota son:
2 Nombre: Soldado 1 | Ubicacion: 6, 4 | Nivel de Vida: 4 | Estado: Vivo | Actitud: Ataque
3
4 Nombre: Soldado 4 | Ubicacion: 4, 3 | Nivel de Vida: 4 | Estado: Vivo | Actitud: Ataque
5
6 Los soldados con el mayor nivel de puntos de la segunda flota son:
```

### Impresión del Promedio de Puntos:

Se imprime el promedio de puntos para la primera y segunda flota.

```
1 El promedio de la primera flota es: 2.75
2 El promedio de la segunda flota es: 3.125
```

### Soldados Ordenados en el Campo de Juego:

Se muestra la información de los soldados ordenados en el campo de juego para ambas flotas.

```
1 SOLDADOS ORDENADOS
2 Soldados ordenados de la primera flota
3 Nombre: Soldado 1 | Ubicacion: 6, 4 | nivelVida: 4 | Estado: Vivo | Actitud: ataque
4 |
5 Nombre: Soldado 2 | Ubicacion: 6, 1 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
6 |
7 Nombre: Soldado 3 | Ubicacion: 6, 8 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
8 |
9 Nombre: Soldado 4 | Ubicacion: 4, 3 | nivelVida: 4 | Estado: Vivo | Actitud: ataque
10 |
11 Nombre: Soldado 5 | Ubicacion: 1, 2 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
12 |
13 Nombre: Soldado 6 | Ubicacion: 5, 7 | nivelVida: 3 | Estado: Vivo | Actitud: ataque
14 |
15 Nombre: Soldado 7 | Ubicacion: 5, 5 | nivelVida: 3 | Estado: Vivo | Actitud: ataque
16 |
17 Nombre: Soldado 8 | Ubicacion: 3, 1 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
18 |
19
20 Soldados ordenados de la segunda flota
21 Nombre: Soldado 1 | Ubicacion: 1, 8 | nivelVida: 4 | Estado: Vivo | Actitud: ataque
22 |
23 Nombre: Soldado 2 | Ubicacion: 7, 7 | nivelVida: 5 | Estado: Vivo | Actitud: ataque
24 |
25 Nombre: Soldado 3 | Ubicacion: 2, 6 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
26 |
27 Nombre: Soldado 4 | Ubicacion: 3, 7 | nivelVida: 1 | Estado: Vivo | Actitud: ataque
28 |
29 Nombre: Soldado 5 | Ubicacion: 5, 2 | nivelVida: 4 | Estado: Vivo | Actitud: ataque
30 |
31 Nombre: Soldado 6 | Ubicacion: 5, 8 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
32 |
33 Nombre: Soldado 7 | Ubicacion: 4, 0 | nivelVida: 5 | Estado: Vivo | Actitud: ataque
34 |
35 Nombre: Soldado 8 | Ubicacion: 5, 9 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
36 |
```

### Ranking de Puntos de Ejército 1 y 2:

Se imprime el ranking de puntos para la primera flota usando el algoritmo de ordenamiento Bubble Sort y para la segunda flota usando el algoritmo de ordenamiento Select Sort.

```
1 RANKING DE PUNTOS EJERCITO 1 POR BUBBLE SORT
2 [Nombre: Soldado 2 | Ubicacion: 6, 1 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
3 , Nombre: Soldado 3 | Ubicacion: 6, 8 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
4 , Nombre: Soldado 5 | Ubicacion: 1, 2 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
5 , Nombre: Soldado 8 | Ubicacion: 3, 1 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
6 , Nombre: Soldado 6 | Ubicacion: 5, 7 | nivelVida: 3 | Estado: Vivo | Actitud: ataque
7 , Nombre: Soldado 7 | Ubicacion: 5, 5 | nivelVida: 3 | Estado: Vivo | Actitud: ataque
8 , Nombre: Soldado 1 | Ubicacion: 6, 4 | nivelVida: 4 | Estado: Vivo | Actitud: ataque
9 , Nombre: Soldado 4 | Ubicacion: 4, 3 | nivelVida: 4 | Estado: Vivo | Actitud: ataque
10 ]
11
12 RANKING DE PUNTOS EJERCITO 2 POR SELECT SORT
13 [Nombre: Soldado 2 | Ubicacion: 7, 7 | nivelVida: 5 | Estado: Vivo | Actitud: ataque
```

```

14 , Nombre: Soldado 7 | Ubicacion: 4, 0 | nivelVida: 5 | Estado: Vivo | Actitud: ataque
15 , Nombre: Soldado 5 | Ubicacion: 5, 2 | nivelVida: 4 | Estado: Vivo | Actitud: ataque
16 , Nombre: Soldado 1 | Ubicacion: 1, 8 | nivelVida: 4 | Estado: Vivo | Actitud: ataque
17 , Nombre: Soldado 3 | Ubicacion: 2, 6 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
18 , Nombre: Soldado 6 | Ubicacion: 5, 8 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
19 , Nombre: Soldado 8 | Ubicacion: 5, 9 | nivelVida: 2 | Estado: Vivo | Actitud: ataque
20 , Nombre: Soldado 4 | Ubicacion: 3, 7 | nivelVida: 1 | Estado: Vivo | Actitud: ataque
21 ]

```

### Inicio del Juego Interactivo:

Se presenta la interfaz del campo de juego con las posiciones de los soldados de ambos equipos.

```

1 #####
2      A  B  C  D  E  F  G  H  I  J
3      -----
4 1 |   |   |   |   | #/5 |   |   |   |   |
5      -----
6 2 |   |   |   | */2 |   |   | */2 |   |   |
7      -----
8 3 |   | */2 |   |   |   | #/4 |   |   |   |
9      -----
10 4 |   |   |   |   | */4 |   |   |   |   |
11      -----
12 5 |   |   |   |   |   |   | */4 |   |   |
13      -----
14 6 |   |   |   |   |   | */3 |   |   |   |
15      -----
16 7 |   |   | #/2 |   |   |   |   |   |   |
17      -----
18 8 |   |   |   | #/1 |   | */3 |   | #/5 |   |
19      -----
20 9 |   | #/4 |   |   |   | #/2 | */2 |   |   |
21      -----
22 10 |   |   |   |   |   | #/2 |   |   |   |
23      -----
24
25 Toca moverse al equipo *
26 Ingrese la posicion de la ficha a mover:

```

### Desarrollo de la Partida Interactiva:

Se muestra la interacción entre los equipos, indicando cuando es el turno de cada uno y solicitando las acciones de movimiento.

```

1 ...
2
3 Toca moverse al equipo *
4 Ingrese la posicion de la ficha a mover:
5 A, 1
6 Jugada no valida
7
8 Toca moverse al equipo *
9 Ingrese la posicion de la ficha a mover:
10 E, 1
11 NO ES EL TURNO DEL EQUIPO: #
12
13 Toca moverse al equipo *
14 Ingrese la posicion de la ficha a mover:
15 D, 2
16 Ingrese la nueva posicion deseada:
17 A, 1
18 Superaste el maximo de casillas por movimiento
19
20 Toca moverse al equipo *

```

```
21 Ingrese la posicion de la ficha a mover:
22 D, 2
23 Ingrese la nueva posicion deseada:
24 C, 2
25 ...
```

### Estado Final del Campo de Juego:

Se muestra el estado final del campo de juego después de las acciones realizadas por los jugadores.

```
1      A   B   C   D   E   F   G   H   I   J
2  -----
3  1 |   |   |   |   | #/5 |   |   |   |   |
4  -----
5  2 |   |   | */2 |   |   |   | */2 |   |   |
6  -----
7  3 |   | */2 |   |   |   | #/4 |   |   |   |
8  -----
9  4 |   |   |   |   | */4 |   |   |   |   |
10 -----
11 5 |   |   |   |   |   |   | */4 |   |   |
12 -----
13 6 |   |   |   |   |   | */3 |   |   |   |
14 -----
15 7 |   |   | #/2 |   |   |   |   |   |   |
16 -----
17 8 |   |   |   | #/1 |   | */3 |   | #/5 |   |
18 -----
19 9 |   | #/4 |   |   |   | #/2 | */2 |   |   |
20 -----
21 10 |   |   |   |   |   |   | #/2 |   |   |   |
22 -----
23
24 Toca moverse al equipo #
25 Ingrese la posicion de la ficha a mover:
```

El juego proporciona una experiencia interactiva, permitiendo al usuario gestionar y mover sus soldados en el campo de juego. Las salidas detalladas en consola ofrecen información clave sobre el estado de las flotas, los soldados y el desarrollo de la partida.

## 11. Commits mas importantes

- A continuacion se muestran los commits mas importantes
- Los commits se hicieron siguiendo la convencion para commits de git, y siguiendo las recomendaciones practicas para hacer mensajes de commits (Mensajes de forma imperativa)

### Commit 7f2adda566d664a4c7c38300c68c287c27e220d6

**Autor:** JhonatanDczel

**Descripción:** Implementa el método "getWinner" que determina el ganador en el juego.

### Commit 22d7b9aac5a07bcc41552fbe65401617f11164db

**Autor:** JhonatanDczel

**Descripción:** Agrega el método para imprimir el ranking de soldados por puntos, proporcionando una visión general del rendimiento de los soldados.

**Commit 883c7b61657b46f8add303a013decb7c71167e3f****Autor:** JhonatanDczel**Descripción:** Introduce métodos para imprimir promedios y puntos, contribuyendo a la visualización estadística de los soldados.**Commit b7aecfc012fda83b168a8a0b318f4e80f528d68f****Autor:** JhonatanDczel**Descripción:** Implementa el método principal "main," que sirve como punto de entrada y coordinación de la aplicación.**Commit 8c95e9be0340d97b729310b1ca0d1ca1a372968e****Autor:** JhonatanDczel**Descripción:** Desarrolla la lógica para modificar un soldado existente en el juego.**Commit 70e2bad4bbd4bddd3d02ce881d90c947d2a302fb****Autor:** JhonatanDczel**Descripción:** Implementa la funcionalidad para eliminar un soldado del ejército.**Commit 496c05baf98c4b3233cbcc33c5b20323e2dba4f9****Autor:** JhonatanDczel**Descripción:** Agrega la funcionalidad para crear un nuevo soldado y añadirlo al ejército.**Commit 89bcc0118d1dd84f6edb69549684e1767a0e77f1****Autor:** JhonatanDczel**Descripción:** Introduce el modo de juego personalizado, proporcionando opciones para personalizar y gestionar los ejércitos.**Commit 274cf853e8a6b713a3ee1a2157103a431cf30009****Autor:** JhonatanDczel**Descripción:** Agrega la funcionalidad inicial, marcando el punto de partida del desarrollo del juego.**Commit 1740eabf3958fed46ecd4be9987d88a37a93d8a2****Autor:** JhonatanDczel**Descripción:** Configura los getters y setters para la clase Soldado, estableciendo la estructura básica de la entidad.**Commit 0c4a358709095cfe75a19e87c1eb852ca20152e1****Autor:** JhonatanDczel**Descripción:** Crea métodos básicos de acciones, sentando las bases para las operaciones relacionadas con los soldados en el juego.



**Commit 5be9489569976460f460a45809a7f17eedf1416a**

**Autor:** JhonatanDczel

**Descripción:** Agrega métodos para el constructor sobrecargado, permitiendo la creación de soldados con parámetros específicos.

**Commit a7fb4575d45eaadc5f8a62a5e5baaa9c956c53a4**

**Autor:** JhonatanDczel

**Descripción:** Introduce atributos para la clase Soldado, estableciendo las propiedades básicas de un soldado en el juego.

**Commit 41f67e32852f4721ae938bbf9d0187685ff05141**

**Autor:** JhonatanDczel

**Descripción:** Crea los archivos para el laboratorio, marcando el inicio del proyecto y la organización de la estructura inicial del código.

## 12. Rúbricas

### 12.1. 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.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
<b>2.0</b>	0.5	1.0	1.5	2.0
<b>4.0</b>	1.0	2.0	3.0	4.0

Tabla 2: Rúbrica para contenido del Informe y demostración

	Contenido y demostración	Puntos	Checklist	Estudiante	Profesor
<b>1. GitHub</b>	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
<b>2. Commits</b>	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
<b>3. Código fuente</b>	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
<b>4. Ejecución</b>	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	1.5	
<b>5. Pregunta</b>	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
<b>6. Fechas</b>	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
<b>7. Ortografía</b>	El documento no muestra errores ortográficos.	2	X	1.5	
<b>8. Madurez</b>	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	
<b>Total</b>		20		19	