

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

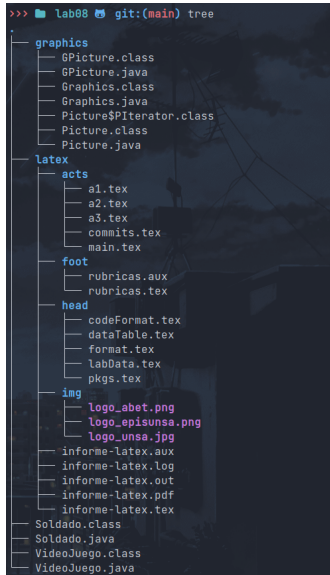
- Sistema Operativo ArchCraft GNU Linux 64 bits Kernell
- NeoVim
- OpenJDK 64-Bit 20.0.1
- Git 2.42.0
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.
- Creacion de programas con CLI
- Biblioteca Graphics (origen propio)

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 lab08

- 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:



5. Inicializando dos ejercitos

- Se necesitan crear dos ejercitos usando HashMap, para lo que crearemos un nuevo método que nos permita hacerlo

```

1 public static HashMap<String, Soldado> initializeArmyHashMap(int n, boolean negro){
2
3     int promLife = 0;
4     Random rand = new Random();
5     int randNum = rand.nextInt(10) + 1;
6     HashMap<String, Soldado> army = new HashMap<>();
7
8     for(int i = 0; i < randNum; i++){
9         String nombre = "Soldado " + n + "x" + i;
10        army.put(nombre, new Soldado(nombre));
11        army.get(nombre).setNegro(negro);
12        army.get(nombre).setLife(rand.nextInt(5) + 1);
13        if(army.get(nombre).getLife() > maxLife.getLife())
14            maxLife = army.get(nombre);
15        promLife += army.get(nombre).getLife();
16        genColumnRow(army.get(nombre));
17    }
18    promLife = promLife / army.size();
19    promedio = (promLife + promedio) / 2;
20    return army;
21
22 }
```

- El código es una adaptación de la generación normal de ejércitos en un array
- Con la diferencia de que los soldados creados estarán conforme en cantidad con el último parámetro que se le pase al método
- Los objetos Soldado se guardan como valores de las claves que son sus nombres
- Los otros dos parámetros que recibe son n (número identificador del ejército) y negro (variable booleana que representa la coloración de un ejército)
- Para acceder a los Soldados y ponerlos se hacen uso de métodos de HashMap
- Se cumple con las especificaciones:
 - Número aleatorio entre 1 y 10
 - Vida aleatoria entre 1 y 5
 - Nombre autogenerado para cada uno
 - Que no hayan dos soldados en una misma casilla, esto se logrará con el siguiente método que sitúa a los soldados sobre el tablero:

```
1 public static void genColumnRow(Soldado s){
2     Random rand = new Random();
3     int column;
4     int row;
5     do {
6         column = rand.nextInt(10);
7         row = rand.nextInt(10);
8     } while(!isEmpty(column, row));
9     s.setColumn(column);
10    s.setRow(row);
11    board[row][column] = s;
12 }
```

- Este código consiste principalmente de un bucle do while
- La condición de parada es que se haya encontrado un sitio vacío para el lugar que se prueba en cada iteración
- Se usa un método auxiliar que devuelve un valor de tipo booleano, es este:

```
1 public static boolean isEmpty(int column, int row){
2     return board[row][column] == null;
3 }
```

- Este código verifica si el espacio en el que queremos insertar un objeto ya está ocupado por otro

6. Mostrando el tablero por pantalla

- Para generar la gráfica nos apoyaremos de la biblioteca graphics, desarrollada el anterior semestre
- Tendremos dos maneras de manejar la gráfica, la primera es un array bidimensional, que contiene las ubicaciones de los soldados en el tablero, y la segunda es un objeto de tipo Picture que contiene la representación gráfica del tablero en un determinado momento

- Para eso necesitaremos dos metodos, el primero de ellos genera un tablero a partir de un array bidimensional (Atributo global)

```
1 public static void makeGBoard(){
2     for(int i = 0; i < 10; i++){
3         Picture fila = null;
4         for(int j = 0; j < 10; j++){
5             Picture c = Picture.casilleroBlanco();
6             if(board[i][j] != null){
7                 c = Picture.soldier().superponer(c);
8                 if(board[i][j].isNegro())
9                     c = Picture.soldier().invertir().superponer(c);
10            }
11            if(j == 0){
12                fila = c;
13                continue;
14            }
15            fila = fila.alLado(c);
16        }
17        if(i == 0){
18            gBoard = fila;
19            continue;
20        }
21        gBoard = gBoard.encima(fila);
22    }
23 }
```

- El metodo itera sobre todos los elementos del array bidimensional "board"
- Con esto contruye el tablero tomando uno a uno sus elementos, en caso de haber un null en cierta posicion, solo imprime un casillero en blanco, en caso de haber un soldado, se pregunta si este tiene coloracion negro, en cuyo caso imprime un soldado negro sobre un fondo blanco, y caso contrario imprime un soldado blanco en casillero blanco
- El segundo metodo que usamos es el que agarra un objeto Picture, y lo grafica:

```
1 public static void displayBoard(){
2     Graphics g = new Graphics(gBoard);
3     g.print();
4 }
```

- El metodo es arto simple, agarra un objeto Picture, genera un nuevo objeto Graphics a partir de el, y lo muestra en pantalla

7. Ordenando un HashMap

- Antes de continuar, las especificaciones como, mostrar al soldado con mayor vida, mostrar el nivel global de vida y los datos por ejercito de todos los soldados ya han sido cubiertas desde muchos laboratorios anteriores, asi que no hay mayor cambio en su funcionamiento
- Asi que ahora nos centraremos en el ultimo cambio que se pide hacer en el laboratorio: el ranking de soldados por vida
- En anteriores laboratorios era demasiado simple implementar un metodo que una dos arrays de Soldados y luego aplicar un algoritmo de ordenamiento por vida

- Ahora las cosas se complican un poco mas, ya que los HashMap no tienen un orden específico
- Para lograr trabajar con el HashMap, lo tendremos que convertir a un array, y para eso usamos el metodo:

```
1 public static Soldado[] toArray(HashMap<String, Soldado> armyH){
2     Soldado[] army = new Soldado[armyH.size()];
3     int i = 0;
4     for(String name : armyH.keySet()){
5         army[i] = armyH.get(name);
6         i++;
7     }
8     return army;
9 }
```

- El codigo es simple, devuelve un array de Soldados, a partir de un HashMap
- Ahora con esto, podemos implementar otro metodo llamado ranking, que tomara dos arrays de Soldados como parametro, los unira, y los ordenara usando un algoritmo de ordenamiento:

```
1 public static HashMap<String, Soldado> ranking(HashMap<String, Soldado> army1, HashMap<String,
2     Soldado> army2){
3     Soldado[] a1 = toArray(army1);
4     Soldado[] a2 = toArray(army2);
5     Soldado[] total = new Soldado[a1.length + a2.length];
6     int i = 0;
7
8     for(Soldado s : a1){
9         total[i] = s;
10        i++;
11    }
12    for(Soldado s : a2){
13        total[i] = s;
14        i++;
15    }
16    bubbleSortLife(total);
17    HashMap<String, Soldado> ranking = new HashMap<>();
18    for(Soldado s : total){
19        ranking.put(s.getName(), s);
20    }
21    return ranking;
22 }
```

- Como vemos, este metodo se apoya del metodo toArray que creamos anteriormente
- Con el metodo ranking ya creado, ahora podemos proceder a mostrar los datos por pantalla

8. Metodo principal

- Ahora procedere a mostrar el metodo main que controla todas las acciones del programa

```
1 public static void main(String[] args){
2     HashMap<String, Soldado> army1 = initializeArmyHashMap(0, false);
3     HashMap<String, Soldado> army2 = initializeArmyHashMap(1, true);
4     displayArmy(army1, "Ejercito 1");
5     displayArmy(army2, "Ejercito 2");
6     System.out.println("Soldado con maxima vida:");
7 }
```

```
7   displaySoldier(maxLife);
8
9   HashMap<String, Soldado> ranking = ranking(army1, army2);
10  displayArmy(ranking, "Ranking de soldados:");
11  makeGBoard();
12  displayBoard();
13
14 }
```

- El metodo inicia generando dos HashMaps
- Luego muestra usando el metodo displayArmy que veremos a continuacion:

```
1  public static void displayArmy(HashMap<String, Soldado> army, String str){
2      System.out.println("\n==== " + str + " =====");
3      for(String soldado : army.keySet()){
4          displaySoldier(army.get(soldado));
5      }
6  }
7
8  public static void displaySoldier(Soldado s){
9      System.out.println(" " + s.getName() + ":");
10     System.out.println(" Nivel de vida: " + s.getLife());
11     System.out.println(" Fila: " + (s.getRow() + 1));
12     System.out.println(" Columna: " + (s.getColumn() + 1));
13     System.out.print("\n");
14 }
```

- El metodo se apoya de otro, (displaySoldier) que muestra los datos de un soldado, y hace eso con todo el array de Soldados
- Continuando con el metodo principal, luego muestra al soldado con la mayor puntuacion de vida
- Luego se genera un nuevo HashMap ranking” que tendra a los soldados ordenados
- Luego llama al metodo ranking e imprime su resultado
- Finalmente se crea el tablero grafico, y se muestra
- Tenemos las siguientes variables globales:

```
1  public static Soldado[] [] board = new Soldado[10][10];
2  public static Picture gBoard;
3  public static Soldado maxLife = new Soldado("sold");
4  public static int promedio = 0;
```

9. Ejecucion grafica y por consola

- A continuacion se vera la Ejecucion tanto grafica como por consola

9.1. Ejecucion por consola

```
1  ===== Ejercito 1 =====
2  Soldado 0x8:
3    Nivel de vida: 3
4    Fila: 9
5    Columna: 2
6
7  Soldado 0x6:
8    Nivel de vida: 3
9    Fila: 3
10   Columna: 1
11
12 Soldado 0x7:
13   Nivel de vida: 5
14   Fila: 1
15   Columna: 1
16
17 Soldado 0x4:
18   Nivel de vida: 4
19   Fila: 2
20   Columna: 2
21
22 Soldado 0x5:
23   Nivel de vida: 1
24   Fila: 6
25   Columna: 3
26
27 Soldado 0x2:
28   Nivel de vida: 4
29   Fila: 4
30   Columna: 6
31
32 Soldado 0x3:
33   Nivel de vida: 3
34   Fila: 10
35   Columna: 1
36
37 Soldado 0x0:
38   Nivel de vida: 4
39   Fila: 2
40   Columna: 6
41
42 Soldado 0x1:
43   Nivel de vida: 2
44   Fila: 3
45   Columna: 3
46
47 ===== Ejercito 2 =====
48 Soldado 1x3:
49   Nivel de vida: 1
50   Fila: 1
51   Columna: 4
52
53
54 Soldado 1x1:
55   Nivel de vida: 3
56   Fila: 5
57   Columna: 3
58
59 Soldado 1x2:
60   Nivel de vida: 4
61   Fila: 3
62   Columna: 9
63
64 Soldado 1x0:
```



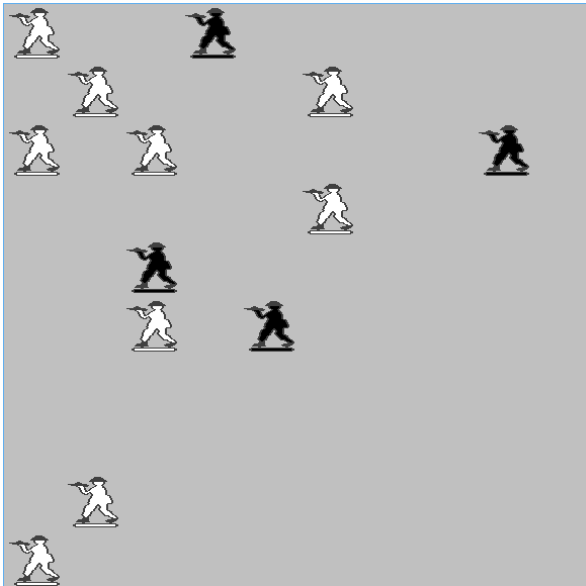
```
65 Nivel de vida: 1
66 Fila: 6
67 Columna: 5
68
69 Soldado con maxima vida:
70 Soldado 0x7:
71 Nivel de vida: 5
72 Fila: 1
73 Columna: 1
74
75
76 ===== Ranking de soldados: =====
77 Soldado 0x7:
78 Nivel de vida: 5
79 Fila: 1
80 Columna: 1
81
82 Soldado 0x2:
83 Nivel de vida: 4
84 Fila: 4
85 Columna: 6
86
87 Soldado 1x2:
88 Nivel de vida: 4
89 Fila: 3
90 Columna: 9
91
92 Soldado 0x0:
93 Nivel de vida: 4
94 Fila: 2
95 Columna: 6
96
97 Soldado 0x4:
98 Nivel de vida: 4
99 Fila: 2
100 Columna: 2
101
102 Soldado 1x1:
103 Nivel de vida: 3
104 Fila: 5
105 Columna: 3
106
107 Soldado 0x8:
108 Nivel de vida: 3
109 Fila: 9
110 Columna: 2
111
112 Soldado 0x6:
113 Nivel de vida: 3
114 Fila: 3
115 Columna: 1
116
117 Soldado 0x3:
118 Nivel de vida: 3
119 Fila: 10
120 Columna: 1
121
122 Soldado 0x1:
123 Nivel de vida: 2
124 Fila: 3
125 Columna: 3
126
127 Soldado 1x3:
128 Nivel de vida: 1
129 Fila: 1
```

```

130 Columna: 4
131
132 Soldado 0x5:
133 Nivel de vida: 1
134 Fila: 6
135 Columna: 3
136
137 Soldado 1x0:
138 Nivel de vida: 1
139 Fila: 6
140 Columna: 5

```

- Como vemos, la estructura del metodo main esta representada en la salida por consola
- A continuacion se vera la salida grafica:



10. 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
- Cada mensaje de commit esta estructurado por un titulo y una descripcion separados por una linea en blanco

Listing 1: commits mas importantes

```

1
2 commit c1082e4d2b725108cdbf1f4acb639a89da4174d8
3 Author: JhonatanDczel <jariasq@unsa.edu.pe>
4 Date: Mon Oct 23 14:02:50 2023 -0500
5
6     Creando el proyecto lab08
7
8     Se copiaron los archivos soldado, videojuego y graphics del laboratorio

```

```
9      anterior
10
11      commit a362ecf01678492c2ba977fd8c3d4f866ccc0313
12      Author: JhonatanDczel <jariasq@unsa.edu.pe>
13      Date: Mon Oct 23 14:12:17 2023 -0500
14
15      Hace que el metodo inicialice army devuelva hashmap
16
17      commit 40035dcc9534d58e9b577105339203b1dd0aa839
18      Author: JhonatanDczel <jariasq@unsa.edu.pe>
19      Date: Mon Oct 23 14:08:29 2023 -0500
20
21      Creacion del metodo inicialiceArmyHashMap
22
23      commit c78508b8e703041b1d11d8410744bed4c9870f31
24      Author: JhonatanDczel <jariasq@unsa.edu.pe>
25      Date: Mon Oct 23 14:25:05 2023 -0500
26
27      Se cambio la forma de establecer el atributo negro
28
29      Se cambio la forma en que se establece que un ejercito sea de color
30      negro, antes del cambio, se usaba un condicional para evaluar el valor
31      booleano negro, ahora, se ingresa directamente este valor como atributo
32      del soldado
33
34      commit 6e7b92190399f9550e32bb58243d0b07ba26d918
35      Author: JhonatanDczel <jariasq@unsa.edu.pe>
36      Date: Mon Oct 23 15:06:45 2023 -0500
37
38      Se crea metodo para ordenar el ejercito
39
40      Como no se puede ordenar directamente un hashmap, ya que no mantiene un
41      oprden especifico, se copia el contenido a un array antes de mandarlo al
42      metodo de ordenamiento de soldados por vida
43
44      commit edfb9c396ca95be5c3038fb18c4bb282894758da
45      Author: JhonatanDczel <jariasq@unsa.edu.pe>
46      Date: Mon Oct 23 15:48:12 2023 -0500
47
48      Adaptando el codigo para hacer el ordenamiento
49
50      Se adapto el codigo para crear un array grande que una a los dos
51      ejercitos, luego se creo el metodo ranking que agarra estos dos
52      ejercitos, los junta y crea uno mas grande que ordena con el algoritmo
53      bubble sort, finalmente el resultado se muestra en pantalla
```

11. Rúbricas

11.1. 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 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 2: 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	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	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	1.5	
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		19	