Informe de Laboratorio 08

Tema: HashMap

Nota

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

Laboratorio	Tema	Duración
08	HashMap	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 18 Octubre 2023	Al 23 Octubre 2023

1. Actividades

- Cree un Proyecto llamado Laboratorio8
- Usted deberá crear las dos clases Soldado.java y VideoJuego5.java. Puede reutilizar lo desarrollado en Laboratorios anteriores.
- Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).
- El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para crear el tablero utilice la estructura de datos más adecuada.
- Tendrá 2 Ejércitos (usar HashMaps). Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados (distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento (indicar conclusiones respecto a este ordenamiento de HashMaps). Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacerlo como programa iterativo.

2. Equipos, materiales y temas utilizados

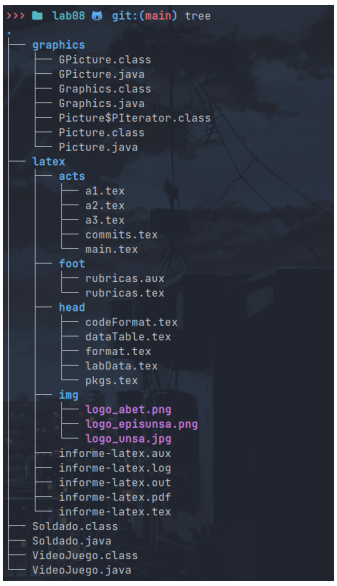
- Sistema Operativo ArchCraft GNU Linux 64 bits Kernell
- NeoVim
- OpenJDK 64-Bit 20.0.1
- Git 2.42.0
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.
- Creacion de programas con CLI
- Bilioteca Graphics (origen propio)

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JhonatanDczel/fp2-23b.git`
- URL para el laboratorio 08 en el Repositorio GitHub.
- `https://github.com/JhonatanDczel/fp2-23b/tree/main/fase02/lab08`
- El trabajo de este laboratorio es en su mayor parte lo mismo que en otros laboratorios, por lo que las variaciones seran minimas

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 usara 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 metodo que nos permita hacerlo

```
public static HashMap<String, Soldado> initializeArmyHashMap(int n, boolean negro){

   int promLife = 0;
   Random rand = new Random();
   int randNum = rand.nextInt(10) + 1;
   HashMap<String, Soldado> army = new HashMap<>();

   for(int i = 0; i < randNum; i++){
     String nombre = "Soldado " + n + "x" + i;
     army.put(nombre,new Soldado(nombre));
     army.get(nombre).setNegro(negro);
     army.get(nombre).setLife(rand.nextInt(5) + 1);
     if(army.get(nombre).getLife() > maxLife.getLife())
       maxLife = army.get(nombre);
     promLife += army.get(nombre).getLife();
     genColumnRow(army.get(nombre));
   }
   promLife = promLife / army.size();
   promedio = (promLife + promedio) / 2;
   return army;

 }
```

- El codigo es una adaptacion de la generacion normal de ejercitos en un array
- Con la diferencia de que los soldados creados estaran conforme en cantidad con el ultimo parametro que se le pase al metodo
- Los objetos Soldado se guardan como valores de las claves que son sus nombres
- Los otros dos parametros que recibe son n (numero identificador del ejercito) y negro (variable booleana que representa la coloracion de un ejercito)
- Para acceder a los Soldados y ponerlos se hacen uso de metodos de HashMap
- Se cumple con las especificaciones:
  - Numero 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 lograra con el siguiente metodo que situa a los soldados sobre el tablero:

```
 public static void genColumnRow(Soldado s){
   Random rand = new Random();
   int column;
   int row;
   do {
     column = rand.nextInt(10);
     row = rand.nextInt(10);
   } while(!isEmpty(column, row));
   s.setColumn(column);
   s.setRow(row);
   board[row][column] = s;
 }
```

- Este codigo consiste principalmente de un bucle do while
- La condicion de parada es que se haya encontrado un sitio vacio para el lugar que se prueba en cada iteracion
- Se usa un metodo auxiliar que devuelve un valor de tipo booleano, es este:

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

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

6. Mostrando el tablero por pantalla

- Para generar la grafica nos apoyaremos de la biblioteca graphics, desarrollada el anterior semestre
- Tendremos dos maneras de manejar la grafica, la primera es un array dibimensional, que contiene las unbicaciones de los soldados en el tablero, y la segunda es un objeto de tipo Picture que contiene la representacion grafica 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)

```
public static void makeGBoard(){
   for(int i = 0; i < 10; i++){
     Picture fila = null;
     for(int j = 0; j < 10; j++){
       Picture c = Picture.casilleroBlanco();
       if(board[i][j] != null){
         c = Picture.soldier().superponer(c);
         if(board[i][j].isNegro())
           c = Picture.soldier().invertir().superponer(c);
       }
       if(j == 0){
         fila = c;
         continue;
       }
       fila = fila.alLado(c);
     }
     if(i == 0){
       gBoard = fila;
       continue;
     }
     gBoard = gBoard.encima(fila);
   }
 }
```

- 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:

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

- 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 especifico
- Para lograr trabajar con el HashMap, lo tendremos que convertir a un array, y para eso usamos el metodo:

```
public static Soldado[] toArray(HashMap<String, Soldado> armyH){
  Soldado[] army = new Soldado[armyH.size()];
  int i = 0;
  for(String name : armyH.keySet()){
    army[i] = armyH.get(name);
    i++;
  }
  return army;
}
```

- 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:

```
public static HashMap<String, Soldado> ranking(HashMap<String, Soldado> army1, HashMap<String,
    Soldado> army2){
  Soldado[] a1 = toArray(army1);
  Soldado[] a2 = toArray(army2);
  Soldado[] total = new Soldado[a1.length + a2.length];
  int i = 0;

  for(Soldado s : a1){
    total[i] = s;
    i++;
  }
  for(Soldado s : a2){
    total[i] = s;
    i++;
  }
  bubbleSortLife(total);
```

```
  HashMap<String, Soldado> ranking = new HashMap<>();
  for(Soldado s : total){
    ranking.put(s.getName(), s);
  }
  return ranking;
}
```

- 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

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

  HashMap<String, Soldado> ranking = ranking(army1, army2);
  displayArmy(ranking, "Ranking de soldados:");
  makeGBoard();
  displayBoard();

}
```

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

```
public static void displayArmy(HashMap<String, Soldado> army, String str){
  System.out.println("\n===== " + str + " =====");
  for(String soldado : army.keySet()){
    displaySoldier(army.get(soldado));
  }
}

public static void displaySoldier(Soldado s){
  System.out.println(" " + s.getName() + ":");
  System.out.println(" Nivel de vida: " + s.getLife());
  System.out.println(" Fila: " + (s.getRow() + 1));
  System.out.println(" Columna: " + (s.getColumn() + 1));
  System.out.print("\n");
}
```

- 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

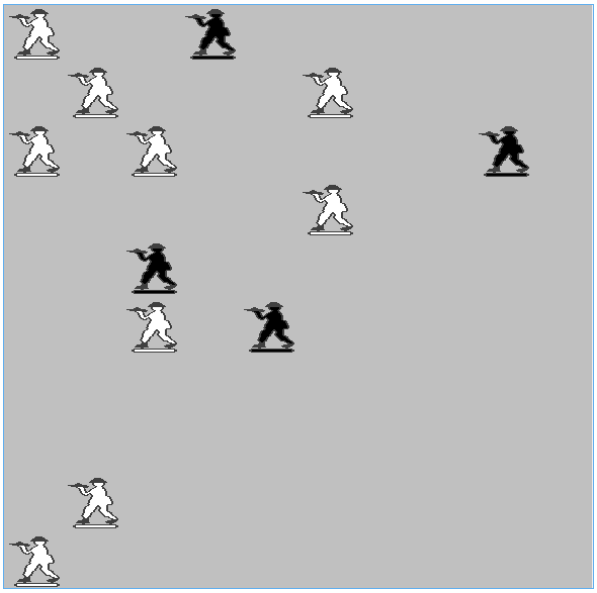
9.1. Ejecucion por consola

```
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
48  ===== Ejercito 2 =====
49   Soldado 1x3:
50    Nivel de vida: 1
51    Fila: 1
52    Columna: 4
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
```

```
Soldado 0x6:
 Nivel de vida: 3
 Fila: 3
 Columna: 1

Soldado 0x3:
 Nivel de vida: 3
 Fila: 10
 Columna: 1

Soldado 0x1:
 Nivel de vida: 2
 Fila: 3
 Columna: 3

Soldado 1x3:
 Nivel de vida: 1
 Fila: 1
 Columna: 4

Soldado 0x5:
 Nivel de vida: 1
 Fila: 6
 Columna: 3

Soldado 1x0:
 Nivel de vida: 1
 Fila: 6
 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

```
commit c1082e4d2b725108cdbf1f4acb639a89da4174d8
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:   Mon Oct 23 14:02:50 2023 -0500

    Creando el proyecto lab08

    Se copiaron los archivos soldado, videojuego y graphics del laboratorio
    anterior

commit a362ecf01678492c2ba977fd8c3d4f866ccc0313
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:   Mon Oct 23 14:12:17 2023 -0500

    Hace que el metodo initialice army devuelva hashmap

commit 40035dcc9534d58e9b577105339203b1dd0aa839
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:   Mon Oct 23 14:08:29 2023 -0500

    Creacion del metodo initialiceArmyHashMap

commit c78508b8e703041b1d11d8410744bed4c9870f31
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:   Mon Oct 23 14:25:05 2023 -0500

    Se cambio la forma de establecer el atributo negro

    Se cambio la forma en que se establece que un ejercito sea de color
    negro, antes del cambio, se usaba un condicional para evaluar el valor
    booleano negro, ahora, se ingresa directemente este valor como atributo
    del soldado

commit 6e7b92190399f9550e32bb58243d0b07ba26d918
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:   Mon Oct 23 15:06:45 2023 -0500

    Se crea metodo para ordenar el ejercito

    Como no se puede ordenar directamente un hashmap, ya que no mantiene un
    oprden especifico, se copia el contenido a un array antes de mandarlo al
    metodo de ordenamiento de soldados por vida

commit edfb9c396ca95be5c3038fb18c4bb282894758da
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:   Mon Oct 23 15:48:12 2023 -0500

    Adaptando el codigo para hacer el ordenamiento

    Se adapto el codigo para crear un array grande que una a los dos
    ejercitos, luego se creo el metodo ranking que agarra estos dos
    ejercitos, los junta y crea uno mas grande que ordena con el algoritmo
    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 cumplio 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 estan 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	