Informe de Laboratorio 07

Tema: Arreglos Estandar y ArrayList

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
07	Arreglos Estandar y ArrayList	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 Laboratorio7
- Usted deberá crear las dos clases Soldado.java y VideoJuego4.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 el tablero utilizar la estructura de datos más adecuada.
- Tendrá 2 Ejércitos (utilizar la estructura de datos más adecuada). 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 y sus puntos de vida (usar caracteres como y otros y 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. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacer el programa iterativo.

2. Equipos, materiales y temas utilizados

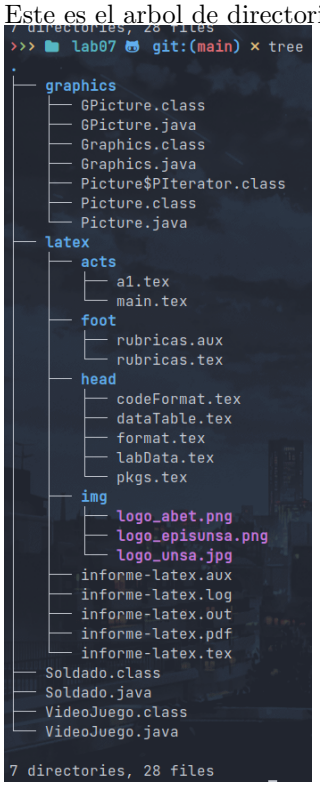
- Sistema Operativo ArchCraft GNU Linux 64 bits Kernell
- NeoVim
- OpenJDK 64-Bit 20.0.1
- Git 2.42.0
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.
- Creacion de programas con CLI
- Bilioteca Graphics (origen propio)

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JhonatanDczel/fp2-23b.git`
- URL para el laboratorio 07 en el Repositorio GitHub.
- `https://github.com/JhonatanDczel/fp2-23b/tree/main/fase02/lab07`

4. Orden de archivos

Este es el arbol de directorios final



5. Actividades iniciales: Crea un proyecto llamado Laboratorio 7

Se creo el directorio lab07, y se copio a el los archivos de los anteriores laboratorios para reciclar partes de codigo

- Se importaron las clases Soldado y Videojuego
- El soldado tiene las especificaciones requeridas como atributos
  - Nombre
  - Puntos de vida
  - Fila
  - Columna
- Para el tablero, se sigue utilizando la biblioteca graphics, la unica indicacion es que se use la estructura de datos mas adecuada, lo que me sugiere que es a libre eleccion, por lo tanto se uso un array bidimensional simple para el tablero

5.1. Regresando el modelo de ArrayList a array simple

- Como dije anteriormente, importamos las anterioes clases de videojuego, y en la clase que importamos, el tablero esta representado con un ArrayList
- Para trabajar mejor se hizo el pase de ArrayList a array simple
- Se logro eso, cambiando los lugares donde trabaja ArrayList

6. Actividad principal

A continuacion se cubriran los requerimientos principales de este laboratorio

6.1. Iniciando 2 ejercitos

En este punto, se inicial dos ejercitos con nombres autogenerados, para eso, se modifica el codigo de inicializacion de ejercitos, para permitir ingresar un parametro extra (Un numero entero que indica el numero de ejercito)

Listing 1: Codigo fuente, Videojuego.java

```
public static void main(String[] args){
   Soldado[] army1 = initializeArmy(0, true);
   Soldado[] army2 = initializeArmy(1, true);
...
  public static void displayArmy(Soldado[] army, String str){
    System.out.println("\n===== " + str + " =====");
    for(Soldado soldier : army){
      displaySoldier(soldier);
    }
  }
```

- Como vemos, los ejercitos se generaran con un numero que sera lo que contendra su nombre

6.2. Modificando los atributos en Soldado.java

Se agrego un nuevo atributo en soldado, que contendra la coloracion del soldado, en este primer caso solo tendra dos opciones de color

- Se agrego el atributo booleano negro
- Se agrego el metodo setter setNegro()
- Se agrego el metodo getter isNegro(), inicialmente se planteo llamarlo getNegro, pero al ser un valor booleano, y siguiendo las buenas practicas de programacion, se cambio a isNegro, para poder trabajar mas facilmente con su valor

Listing 2: Codigo fuente, Videojuego.java

```
public class Soldado{
  public String name;
  public int life;
  public int row;
  public int column;
  public boolean negro = false;

...
...
...


  //SECCION DE SETERS

  public void setNegro(boolean n){
    this.negro = n;
  }

...
...
...

  //SECCION DE GETERS

  public boolean isNegro(){
    return this.negro;
  }
```

6.3. Implementacion del color del ejercito

- Una vez que ya tenemos una forma de distinguir un soldado, se pueden colorear de diferentes tonos
- Para eso se usa el metodo invertir(), incluido en graphics, que cambia o invierte el color de una ficha/pieza
- Al momento de inicializar un ejercito se tendra que especificar el color del mismo

Listing 3: Codigo fuente, Videojuego.java

```

  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);
      }
```

- En esa porcion de codigo, vemos el metodo que dibuja a los soldados por pantalla
- Se verifica si alguno tiene el metodo de isNegro como true, en caso de ser cierto, se invierte el color de un soldado y se agrega al tablero

6.4. Muestra de datos por consola

- La parte final de la actividad es basicamente lo que hicimos por 3 laboratorios seguidos, mostrar los datos de vida, ordenar por algun criterio, generar un ranking de soldados y mostrar sus niveles de vida asi como el promedio y la vida mas grande
- Por lo tanto solo mostraremos el codigo sin explayarnos en su funcionamiento

Listing 4: Codigo fuente, Videojuego.java

```
public static void main(String[] args){
  Soldado[] army1 = initializeArmy(0, true);
  Soldado[] army2 = initializeArmy(1, true);
  displayArmy(army1, "Ejercito 1");
  displayArmy(army2, "Ejercito 2");
  System.out.println("Soldado con maxima vida:");
  displaySoldier(maxLife);
  bubbleSortLife(army1);
  displayArmy(army1, "Ranking de soldados:");
  makeGBoard();
  displayBoard();
}
```

- El codigo pertenece al metodo principal
- Se inicializan dos ejercitos de soldados
- Se muestran por pantalla, gracias al metodo displayArmy, que mostrara sus niveles de vida, filas, columnas, nombres y puntos
- Se muestra el soldado de mayor vida, usando el metodo que hicimos dos laboratorios atras
- Se muestra el ranking de soldados, usando el metodo que une a los dos ejercitos, los ordena y muestra el resputado por pantalla
- Se utiliza el algoritmo de ordenamiento bubble sort para ordenar los ejercitos
- y finalmente se muestra el tablero gracias al metodo makeBoard y DisplayBoard que crea, y muestra el tablero respectivamente

7. Ejecucion

- A continuacion mostraremos la ejecucion de Videojuego.java en sus dos formas, la forma por consola, y la forma grafica

7.1. Consola

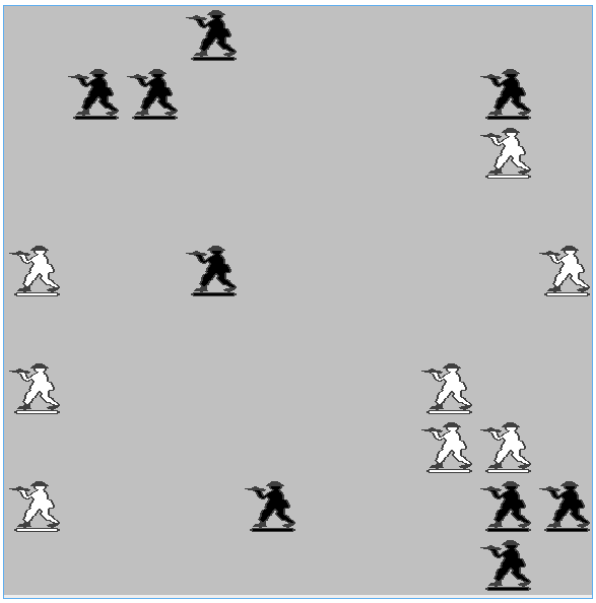
Listing 5: Ejecucion del codigo por consola

```

===== Ejercito 1 =====
 Soldado 0x0:
  Nivel de vida: 3
  Fila: 9
  Columna: 1

 Soldado 0x1:
  Nivel de vida: 2
  Fila: 8
  Columna: 9

 Soldado 0x2:
  Nivel de vida: 5
  Fila: 3
  Columna: 9

 Soldado 0x3:
  Nivel de vida: 3
  Fila: 5
  Columna: 10

 Soldado 0x4:
  Nivel de vida: 1
  Fila: 7
  Columna: 1

 Soldado 0x5:
  Nivel de vida: 2
  Fila: 7
  Columna: 8

 Soldado 0x6:
  Nivel de vida: 4
  Fila: 8
  Columna: 8

 Soldado 0x7:
  Nivel de vida: 1
  Fila: 5
  Columna: 1


===== Ejercito 2 =====
 Soldado 1x0:
  Nivel de vida: 5
  Fila: 2
  Columna: 3

 Soldado 1x1:
  Nivel de vida: 1
  Fila: 10
  Columna: 9

 Soldado 1x2:
  Nivel de vida: 2
  Fila: 9
  Columna: 5

 Soldado 1x3:
  Nivel de vida: 3
```

```
 Fila: 2
 Columna: 9

Soldado 1x4:
 Nivel de vida: 4
 Fila: 1
 Columna: 4

Soldado 1x5:
 Nivel de vida: 1
 Fila: 2
 Columna: 2

Soldado 1x6:
 Nivel de vida: 4
 Fila: 9
 Columna: 9

Soldado 1x7:
 Nivel de vida: 1
 Fila: 5
 Columna: 4

Soldado 1x8:
 Nivel de vida: 3
 Fila: 9
 Columna: 10

Soldado con maxima vida:
 Soldado 0x2:
  Nivel de vida: 5
  Fila: 3
  Columna: 9


===== Ranking de soldados: =====
 Soldado 0x4:
  Nivel de vida: 1
  Fila: 7
  Columna: 1

 Soldado 0x1:
  Nivel de vida: 2
  Fila: 8
  Columna: 9

 Soldado 0x5:
  Nivel de vida: 2
  Fila: 7
  Columna: 8

 Soldado 0x7:
  Nivel de vida: 1
  Fila: 5
  Columna: 1

 Soldado 0x0:
  Nivel de vida: 3
  Fila: 9
  Columna: 1

 Soldado 0x3:
  Nivel de vida: 3
  Fila: 5
  Columna: 10
```

```
Soldado 0x6:
 Nivel de vida: 4
 Fila: 8
 Columna: 8

Soldado 0x2:
 Nivel de vida: 5
 Fila: 3
 Columna: 9
```

- A continuacion mostraremos la salida grafica del codigo



8. Commits mas importantes

- A continuacion se muestran los commits mas importantes

Listing 6: commits mas importantes

```
commit 53bc5c08d38fb9d931979bee8b1d013a615d7384
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Wed Oct 18 11:54:46 2023 -0500

    Actividad 1: Crea un proyecto llamado Laboratorio7

    Se creo el directorio lab07 como subdirectorio de fase02

commit 87e61fbcfd20fe0660ae218f3f324e23e5166329
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Wed Oct 18 12:07:32 2023 -0500

    Actividad4: Regresa el modelo de ArrayList a array

    Siguiendo las indicaciones, se escogio el modelo de array bidimensional
    simple para el tablero, ya que el modo de trabajar con el es mas simple,
    la forma de acceder a elementos, modificarlos y crearlos es mucho mas
```

```
    rapido y eficiente que con un ArrayList.commit 94fbd5984ef78bba0f3279931da37cdb9
ae225f7


commit 94fbd5984ef78bba0f3279931da37cdb9ae225f7
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:   Wed Oct 18 12:17:40 2023 -0500

    Actividad5: Inicia 2 ejercitos

    En este punto, se inicial dos ejercitos con nombres autogenerados, para
    eso, se modifica el codigo de inicializacion de jeercitos, para permitir
    ingresar un parametro extra (Un numero entero que indica el numero de
    ejercito)

commit 5c63eb95e23f2878921d8c0e2ddae6b430c5df17
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:   Wed Oct 18 12:20:37 2023 -0500

    Actividad5: Modifica los tributos en Soldado.java

    Se agrega un atributo llamado "negro" que es de valor booleano, es true
    en caso de querer que nuestro ejercito se muestre de color negro, y
    false, si queremos que nuestro ejercito se muestre en el color por
    defecto, blanco.

commit da60e154a436ff0c1b4411386f323a14402edf6e
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:   Wed Oct 18 12:35:23 2023 -0500

    Actividad 5: Implementa el cambio de color

    Se implemento el codigo en el metodo makeGBoard, para invertir el color
    en caso de que el atributo "isNegro" sea verdadero, para invertir el
    color se usa un metodo que esta en la biblioteca graphcis, que cambia la
    forma en que es representada la figura y luego cambia tambien el color
    en que es dibujada la figura


commit da532908095e179fce9779e9963d13762d536b6a
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:   Wed Oct 18 12:37:20 2023 -0500

    Actividad5: Prueba de coloreado

    Se hicieron 3 pruebas de coloreado de soldados, la primera con los
    soldados en blanco, la segunda alternando colores, y la tercera con los
    soldados de color
```

9. Rúbricas

9.1. 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	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 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	3	
Total		20		19	