Informe de Laboratorio 06

Tema: Array List

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
06	Array List	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 12 Octubre 2023	Al 15 Octubre 2023

1. Actividades

- Cree un Proyecto llamado Laboratorio6
- Tendrá 2 Ejércitos. 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. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla).

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 06 en el Repositorio GitHub.
- `https://github.com/JhonatanDczel/fp2-23b/tree/main/fase02/lab06`

4. Disclaimer

- El laboratorio 06 es una leve modificacion al laboratorio 05
- Siguiendo la instruccion de poder reutilizar codigo de otros laboratorios, el trabajo practicamente esta hecho en un 70
- Para seguir el historial de modificaciones, utilizaremos los commits como referencia
- Los commits fueron hechos con el enfoque de las 7 reglas de los buenos commits (La convencion que sugiere el mismo git), que consta de un titulo y un cuerpo
- El titulo contiene, en forma imperativa, la accion que se llevara a cabo en cada uno de los commits, esta en modo imperativo para responder a la pregunta .Este commit, si se aplica, hace.."
- El body contiene detalles sobre los cambios introducidos en el commit
- Dada la naturaleza de los commit como reportes, el presente informe de laboratorio se podra detallar completamente con los mensajes del body, agregare comentarios unicamente donde sean pertinentes

5. Cambiando modelo a ArrayList

- La primera modificacion viene sobre el uso de arrays bidimensaionales, para cambiarlo a usar ArrayList bidimensionales, para esto tendremos que tomar en cuenta la forma de introducir y obtener datos de un ArrayList

Listing 1: Commit principal

```
commit 394799acf85f7bbde78257b4c2deb3d9a90b6674
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 15:58:14 2023 -0500

    Lab 06: Cambiando modelo a ArrayList

    Dado que la unica expecificaciones que el tablero sea un arraylist, es
    unt rabajo trivial cambiar de tipo de lista y cambiar tambien la forma
    en que se accede a los elementos y como se setean elementos
```

- El principal cambio que se hizo, fue declarar board como un array list bidimensaional de tamanho 10, luego se recorre el ArrayList para crear otros 10 ArrayList dentro de ellos, generando asi las filas y columnas

Listing 2: VideoJuego.java

```
public static ArrayList<ArrayList<Soldado>> board = new ArrayList<>(10);
public static Picture gBoard;
public static Soldado maxLife = new Soldado("sold");
public static int promedio = 0;

public static void main(String[] args){
  for (int i = 0; i < 10; i++) {
    ArrayList<Soldado> fila = new ArrayList<>(10);
    for (int j = 0; j < 10; j++) {
      fila.add(null);
    }
    board.add(fila);
  }
```

6. Creando dos ejercitos

Listing 3: Commit principal

```
commit d4c878a641ecad05ce16390458da3bf0d0018fd3
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 16:02:24 2023 -0500

   Crea dos ejercitos de soldados y los muestra

   Se crean dos ejercitos, seteando sus nombres con "1x1"... luego se
   ubican en pantalla y los datos de creacion se muestran por consola
```

- Los cambios se veran a continuacion:

Listing 4: VideoJuego.java

```
   Soldado[] army1 = initializeArmy(1);
   Soldado[] army2 = initializeArmy(2);

   displayArmy(army1);
   displayArmy(army2);
...
...
...
 public static Soldado[] initializeArmy(int n){
   int promLife = 0;
   Random rand = new Random();
   int randNum = rand.nextInt(10) + 1;
   Soldado[] army = new Soldado[randNum];

   for(int i = 0; i < randNum; i++){
     army[i] = new Soldado("Soldado " + n + "x" + (i + 1));
     army[i].setLife(rand.nextInt(5) + 1);
     if(army[i].getLife() > maxLife.getLife())
       maxLife = army[i];
     promLife += army[i].getLife();
     genColumnRow(army[i]);
```

```
    }
    promLife = promLife / army.length;
    promedio = (promLife + promedio) / 2;
    return army;
  }
```

- El cambio fue en la manera en que se agregan los nombres, para adecuarlos a 1x0.. 1x2...

7. Ranking de soldados por vida

Listing 5: Commit principal

```
commit 6316d6416b04735eefb840dc9898c7be70712a4c
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 16:11:06 2023 -0500

    Agregar ranking de soldados por vida

    Se agrego la funcionalidad de hacer un ranking de soldados, para eso, se
    juntan los dos ejercitos y luego se aplica el ordenamiento por bubble
    sort
```

- El ranking se hizo usando bubble sort como algoritmo de ordenamiento
- El segundo algoritmo de ordenamiento que se utiliza es insertion sort
- Como tenemos dos ejercitos, se crea un nuevo array uniendo a los dos, y luego se aplica el ordenamiento

Listing 6: VideoJuego.java

```
  public static void insertionSortLife(Soldado[] army){
    for(int i = 1; i < army.length; i++){
      Soldado actual = army[i];
      int j = i - 1;
      while(j >= 0 && army[j].getLife() > actual.getLife()){
        army[j + 1] = army[j];
        j--;
      }
      army[j + 1] = actual;
    }
  }

  public static void bubbleSortLife(Soldado[] army){
    for(int i = 0; i < army.length - i; i++){
      for(int j = 0; j < army.length - 1 - i; j++){
        if(army[j].getLife() > army[j + 1].getLife())
          intercambiar(army, j, j + 1);
      }
    }
  }
  public static void intercambiar(Soldado[] army, int i, int j){
    Soldado aux = army[i];
    army[i] = army[j];
    army[j] = aux;
  }
}
```

- Se usan 3 metodos, uno auxiliar para intercambiar, y otros dos que son Bubble e Insertion sort

8. Implementar forma de encontrar un ganador

Listing 7: Commit principal

```
commit bad9e0d93b8adc82891638530b51dc0250daa7d1
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 16:17:23 2023 -0500

    Implementa mecanismo para determinar ganador

    Se implemento el metodo WhoWins, que detecta cual de los dos ejercitos
    resulto como ganador, para esto se basa en la cantidad de soldados que
    tiene cada ejercito, ya que siempre se impondra la cantidad numerica en
    una batalla
```

- Como ya esta mencionado, la forma de encontrar un ganador es comparando la cantidad numerica de soldados

9. Adicional: Distintos colores, distintos ejercitos

- Para aplicar un atractivo visual, y para distinguir entre ejercitos, vamos a implementar en el codigo la funcionalidad de especificar si queremos que un ejercito sea de color negro

Listing 8: Commit principal

```
commit 4b5eecad1cf099cf55b1e66815fc5c8cf952848c (HEAD -> main, origin/main)
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 16:43:16 2023 -0500

    Implementacion de distinto color para ejercitos

    Se agrego una moficiacion en Videojuego.java, para pintar de negro
    ciertos ejercitos y asi tener dos colores para representarlos
    El Soldado.java se agrego un nuevo atributo, que indica si un soldado
    debera colorearse de negro o no
```

- Para eso primero se modifica la clase soldado
- Agregamos un nuevo atributo
- Agregamos metodos getters
- Agregamos metodos setters

Listing 9: Soldado.java

```
public class Soldado{
...
...
  public boolean negro = false;

  public Soldado(String name){
    this.name = name;
  }

  //SECCION DE SETERS

```

```
12   public void setNegro(boolean n){
13     this.negro = n;
14   ...
15   ...
16
17   //SECCION DE GETERS
18
19  ...
20  ...
21
22   public boolean getNegro(){
23     return this.negro;
24   }
25
26  ...
27  ...
28
29  }
```

- Ahora se agrega una sentencia if a nuestro metodo makeGBoard, que genera el tablero grafico

Listing 10: Soldado.java

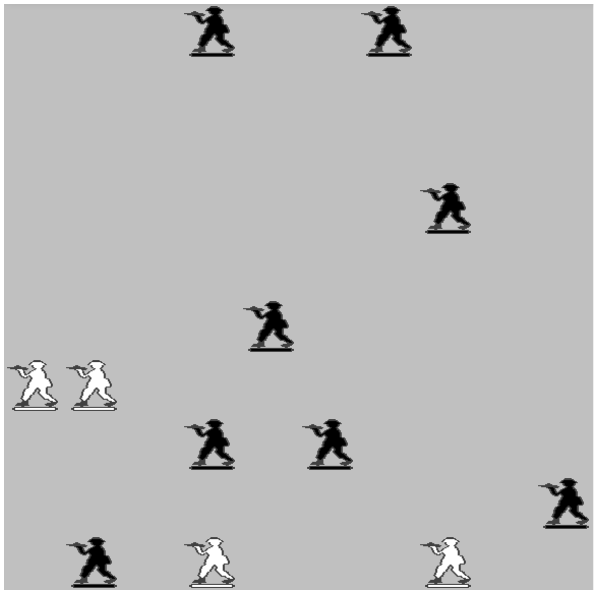
```
1   ...
2   ...
3
4         Picture c = Picture.casilleroBlanco();
5         if(board.get(i).get(j) != null){
6           Picture sold = Picture.soldier();
7           if(board.get(i).get(j).getNegro())
8             sold = sold.invertir();
9           c = sold.superponer(c);
10        }
11
12  ...
13  ...
```

- Con esta condicional, en caso de que el atributo negro de Soldado.java sea true, se invertira el color del soldado
- Para invertir el color, se usa el metodo de la clase Picture, en la biblioteca graphics

10. Ejecucion

- A continuacion veremos un ejemplo de la simulacion de batalla
- Tenemos dos salidas, una grafica y otra por consola
- La grafica por consola muestra:
  - Los datos de los soldados de los ejercitos creados
  - Los datos del soldado con mas puntos
  - El ranking de soldados tomando como referencia la cantidad de vida que poesen
  - El ganador de la batalla tomando en cuenta el factor de la cantidad numerica de soldados en ambos ejercitos
- La grafica del campo de batalla contiene a los soldados que han sido creados, cada uno ubicado en su posicion en filas y columnas

10.1. Grafica del campo



- Cada soldado corresponde a uno que ha sido creado, y esta ubicado en la misma posicion que lo que indica la consola

10.2. Grafica por consola

- A continuacion veremos la salida de consola:

Listing 11: Ejecucion por consola

```
===== Ejercito 1 =====
 Soldado 1x1:
  Nivel de vida: 2
  Fila: 1
  Columna: 7

 Soldado 1x2:
  Nivel de vida: 4
  Fila: 1
  Columna: 4

 Soldado 1x3:
  Nivel de vida: 2
  Fila: 4
  Columna: 8

 Soldado 1x4:
  Nivel de vida: 5
  Fila: 8
  Columna: 4

 Soldado 1x5:
  Nivel de vida: 1
  Fila: 6
  Columna: 5

 Soldado 1x6:
  Nivel de vida: 3
```

```
 Fila: 10
 Columna: 2

 Soldado 1x7:
  Nivel de vida: 1
  Fila: 8
  Columna: 6

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


===== Ejercito 2 =====
 Soldado 2x1:
  Nivel de vida: 3
  Fila: 10
  Columna: 4

 Soldado 2x2:
  Nivel de vida: 2
  Fila: 7
  Columna: 2

 Soldado 2x3:
  Nivel de vida: 5
  Fila: 7
  Columna: 1

 Soldado 2x4:
  Nivel de vida: 4
  Fila: 10
  Columna: 8

Soldado con maxima vida:
 Soldado 1x4:
  Nivel de vida: 5
  Fila: 8
  Columna: 4

Ranking de soldados por vida:

===== =====
 Soldado 1x5:
  Nivel de vida: 1
  Fila: 6
  Columna: 5

 Soldado 1x7:
  Nivel de vida: 1
  Fila: 8
  Columna: 6

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

 Soldado 1x1:
  Nivel de vida: 2
  Fila: 1
  Columna: 7

 Soldado 1x3:
```

```
 Nivel de vida: 2
 Fila: 4
 Columna: 8

Soldado 2x2:
 Nivel de vida: 2
 Fila: 7
 Columna: 2

Soldado 1x6:
 Nivel de vida: 3
 Fila: 10
 Columna: 2

Soldado 2x1:
 Nivel de vida: 3
 Fila: 10
 Columna: 4

Soldado 1x2:
 Nivel de vida: 4
 Fila: 1
 Columna: 4

Soldado 2x4:
 Nivel de vida: 4
 Fila: 10
 Columna: 8

Soldado 1x4:
 Nivel de vida: 5
 Fila: 8
 Columna: 4

Soldado 2x3:
 Nivel de vida: 5
 Fila: 7
 Columna: 1

La metrica tomada para el ganador es: cantidad

***** Army 1 is the winner! *****
```

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	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	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	3	
Total		20		18.5	