Informe de Laboratorio 05

Tema: Arreglos bidimensionales de objetos

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
05	Arreglos bidimensionales de objetos	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 Laboratorio5
- Usted deberá crear las dos clases Soldado.java y VideoJuego2.java. Puede reutilizar lo desarrollado en Laboratorio 3 y 4.
- 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. Pero ahora el tablero debe ser un arreglo bidimensional de objetos.
- Inicializar el tablero con n soldados aleatorios entre 1 y 10. Cada soldado tendrá un nombre autogenerado: Soldado0, Soldado1, 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 (usar caracteres como | y otros). Además de los datos del Soldado con mayor vida, el promedio de puntos de vida de todos los soldados creados, el nivel de vida de todo el ejército, los datos de todos los soldados en el orden que fueron creados y un ranking de poder de todos los soldados creados, del que tiene más nivel de vida al que tiene menos (usar al menos 2 algoritmos de ordenamiento).

2. Equipos, materiales y temas utilizados

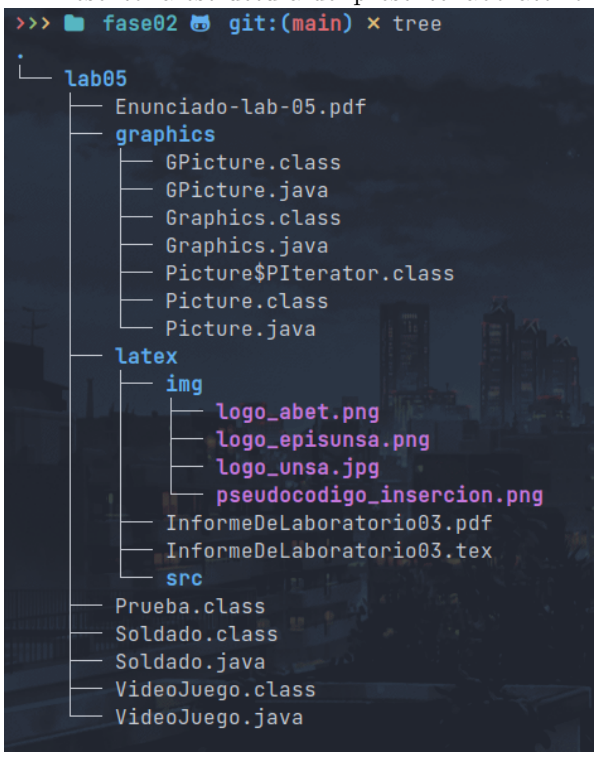
- Sistema Operativo ArchCraft GNU Linux 64 bits Kernell
- NeoVim
- OpenJDK 64-Bit 20.0.1
- Git 2.42.0
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.
- Creacion de programas con CLI
- Bilioteca Graphics (origen propio)g

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JhonatanDczel/fp2-23b.git`
- URL para el laboratorio 05 en el Repositorio GitHub.
- `https://github.com/JhonatanDczel/fp2-23b/tree/main/fase02/lab05`

4. Actividad 1

Presento la estructura del presente laboratorio:



5. Actividad 2 y 3

- Para esta actividad de creo la clase soldado con los siguientes atributos

Listing 1: Clase Soldado.java

```java
public class Soldado{
  public String name;
  public int life;
  public int row;
  public int column;

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

- Cada uno de los atributos tiene sus apropiados metodos setters y getters
- La actividad 4 en realidad solo es una especificacion asi que continuemos con la actividad 5

6. Actividad 5

- Se modifico el codigo de videojuego.java del laboratorio 3, para hacer que la inicializacion dejercitos nos de como resultado un ejercito de entre 1 y 10 soldados
- Para esto usamos la clase Random

Listing 2: Clase Videojuego.java

```java
  public static Soldado[] initializeArmy(){
    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 " + (i + 1));
    }
    return army;
  }
```

- Ahora agregaremos la funcionalidad de que cada soldado se genere con un nivel de vida de entre 1 y 5

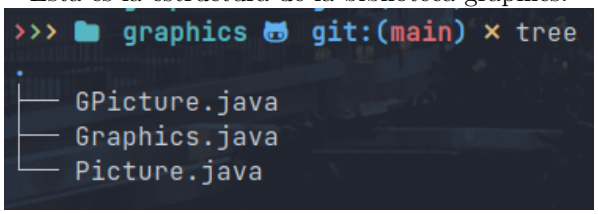
Listing 3: Clase Videojuego.java

```java
  public static Soldado[] initializeArmy(){
    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 " + (i + 1));
      army[i].setLife(rand.nextInt(5) + 1);
    }
```

Ahora, para graficar a los soldados usaremos una biblioteca que fue desarrollada el semestre pasado en Fundamentos de la Programacion 1. Esta biblioteca es graphics, que puede mostrar un grafico de un tablero de ajedrez por pantalla, para mostrar a los soldados se creo un nuevo tipo de figura (soldier)

Esta es la estructura de la biblioteca graphics:



- Lo que se hizo a continuacion es un metodo que grafique el estado actual del tablero, usando graphics

Listing 4: Clase Videojuego.java

```
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(j == 0){
9          fila = c;
10         continue;
11       }
12       fila = fila.alLado(c);
13     }
14     if(i == 0){
15       gBoard = fila;
16       continue;
17     }
18     gBoard = gBoard.encima(fila);
19   }
20 }
```

- Se recorre por filas y columnas sobre los elementos de nuestro ejercito de soldados, y en caso de encontrar a un soldado en alguna posicion (boardi[i][j] != null) imprime un soldado en esa posicion y la manda a una fila general que la mandara a su vez a un tablero (gBoard)
- Al final del metodo tendremos el tablero gBoard con los soldados en las posiciones requeridas

6.1. Mostrando los datos por pantalla

- Ahora que tenemos todo lo necesario para correr el juego, necesitamos metodos para mostrar todo lo que hemos trabajado
- Tenemos 3 display's, mostrar un soldado, un ejercito, y el tablero, veamos el primero:

6.1.1. Mostrar soldado

Listing 5: Clase Videojuego.java

```
1  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");
 }
```

- Como vemos, el metodo recibe un soldado, y luego muestra datos como la vida, el nombre, la fila en la que se ubica y la columna en la que esta
- Este metodo nos vservira por que lo usaremos despues en el metodo de mostrar ejercito y tambien para mostrar al soldado con mayor puntaje de vida

6.1.2. Mostrar un ejercito

Listing 6: Clase Videojuego.java

```
public static void displayArmy(Soldado[] army){
  System.out.println("\n===== Army Soldiers =====");
  for(Soldado soldier : army){
    displaySoldier(soldier);
  }
}
```

- La entrada que recibe el metodo es un ejercito de soldados
- El metodo recorre el ejercito entero
- En cada vuelta, hace una lolamada al metodo que desarrollamos antes

6.1.3. Mostrando el tablero en pantalla

- Quiza esto es un poco mas facil
- Usamos la clase Graphics de la biblioteca graphics, para imprimir en una ventana emergente el tablero que ya creamos con anterioridad:

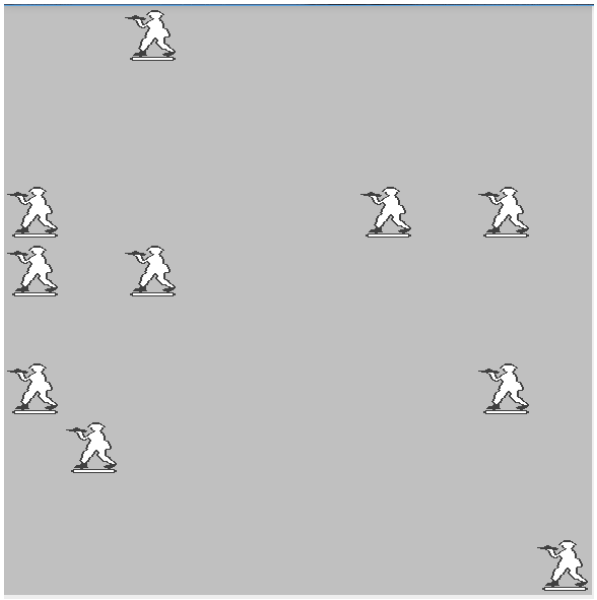
Listing 7: Clase Videojuego.java

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

6.1.4. Ejecucion

- La ejecucion de esta porcion de codigo esta dividida en dos partes
- La primera parte es la parte grafica, el tablero se dibuja en la pantalla y muestra a los soldados que han sido creados
- La segunda parte es la ista en consola, que muestra los datos de los sondados que han sido creados
- Adicionalmente se pide el soldado con mas vidas, y el promedio de vidas

- Esto lo lograremos poniendo contadores que recopilen datos y usamos los metodos de display para mostrarlos en pantalla:



- Ahora veremos los datos mostrados por consola:

Listing 8: Consola

```
===== Army Soldiers =====
Soldado 1:
 Nivel de vida: 5
 Fila: 4
 Columna: 9

Soldado 2:
 Nivel de vida: 5
 Fila: 4
 Columna: 7

Soldado 3:
 Nivel de vida: 5
 Fila: 1
 Columna: 3

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

Soldado 5:
 Nivel de vida: 2
 Fila: 10
 Columna: 10

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

Soldado 7:
 Nivel de vida: 5
```

```
 Fila: 7
 Columna: 1

Soldado 8:
 Nivel de vida: 3
 Fila: 7
 Columna: 9

Soldado 9:
 Nivel de vida: 5
 Fila: 8
 Columna: 2

Soldado 10:
 Nivel de vida: 1
 Fila: 5
 Columna: 3

Soldado con maxima vida:
Soldado 1:
 Nivel de vida: 5
 Fila: 4
 Columna: 9
```

6.2. Orednamiento por vida

- Para hacer el ordenamiento de los soldados por dos metodos de ordenamiento, lo haremos por buble sort y por insertion sort
- Como se nos dio la indicacion que podemos reutilizar el codigo de los laboratorios 3 y 4, usaremos la implementacion de algoritmos de ordenamiento del laboratorio 4
- Con esta gran ayuda, la implementacion de ordenamiento por vida se reduce a reemplazar el nombre de las variables y clases usadas

6.2.1. Bubble sort

- Costa de dos metodos, el ordenamiento es una implementacion del algoritmo buble sort
- El metodo auxiliar para intercambiar dos elementos de un ejercito

Listing 9: Metodo auxiliar

```java
public static void intercambiar(Soldado[] army, int i, int j){
  Soldado aux = army[i];
  army[i] = army[j];
  army[j] = aux;
}
```

Listing 10: Metodo bubble sort

```java
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);
    }
  }
}
```

6.2.2. Insertion sort

- El metodo de ordenamiento por vida usando el algoritmo de insertion sort solo consta de un metodo principal

Listing 11: Metodo bubble sort

```
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;
  }
}
```

7. Commits

- Estuve investigando sobre buenas practicas de redaccion de commits y encontre las 7 reglas de los mensajes de commits
- Estas reglas fueorn aplicadas en cada uno de los commits
- De manera que no es necesario hacer una explicacion a detalle de cada uno, ya que la explicacion la contiene el mismo mensaje de commit
- A continuacion muestro los commits principales:

Listing 12: Commits principales

```
commit d11009b95725f37e26823da21d9479c457c44c70
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Thu Oct 12 15:10:27 2023 -0500

   Crear el proyecto laboratorio 05

   Agregar el documento de enunciado y crear la estructura de
   archivos para el trabajo
```

Listing 13: Commits principales

```
commit f1ef7a39016c465a391cce3804443090edc9cc94
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 09:24:41 2023 -0500

   Completa la actividad 2

   Se creo la clase soldado.java con las especificaciones requeridas de
   atributos y metodos
```

Listing 14: Commits principales

```
commit 5a4c515463e334555acd69dadea0b3bfb40d321d
Author: JhonatanDczel <jariasq@unsa.edu.pe>
```

```
Date:  Mon Oct 16 09:34:44 2023 -0500

    Avanza la actividad 5

    Se puso el sistema para que el soldado reciba una cantidad de vida
    aleatoriamente entre 1 y 5
```

Listing 15: Commits principales

```
commit d53d0fb5620edef79329eb845833aea3df29641d
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 10:11:06 2023 -0500

    actividad 5: biblioteca para graficar el tablero

    Se agrego la biblioteca graphics.jar que hicimos en Fundamentos 1, para
    graficar el tablero con los soldados representados por peones
```

Listing 16: Commits principales

```
commit 21c0de14822b4271d21dc2df144d7626f7398eb1
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 13:17:11 2023 -0500

    Modificar Picture para mostrar soldados

    Mosifique el codigo de la biblioteca graphics para generar en pantalla
    un soldado
```

Listing 17: Commits principales

```
commit 686cc633b17c5967511b97dc8ef0d311adf6d12a
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 13:38:43 2023 -0500

    Crea el etodo displayBoard

    este metodo agarra el tablero (gBoard) y abre una ventana con la
    representacion grafica del estado actual
```

Listing 18: Commits principales

```
commit d162a8925fe59ea70ed1102a2c765a9f4bdd78ad
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 13:50:32 2023 -0500

    Agrega el prototipo para calcular la vida promedio

    Se agrego codigo para recabar datos en la inicializacion de los
    ejercitos, luego dividirlos entre el total de soldados y finalmente
    guardarlos en una variable que contiene el promedio de vida de los ejercito
```

Listing 19: Commits principales

```
commit 7a9b21503e5bb29321cf14cf63e80a516022a664
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 13:58:11 2023 -0500

    Reconoce al soldado de mayor vida

```

```
   En la inicializacion de ejercitos, se agrega codigo para comparar los
   nuevos soldados que van siendo creados con el soldado actual con mayor
   vida, en caso de que uno de ellos tenga mas vidas que este, se convierte
   en el nuevo soldado con mayor vida
```

Listing 20: Commits principales

```
commit 91add39b83886415a2102efd661401efd5290e83
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 14:18:59 2023 -0500

   Implementa el metodo de ordenamiento bubble

   Se Agregan dos metodos, uno auxiliar para intercambiar dos soldados, y
   otro que es el metodo de ordenamiento por burbuja, usanco como base de
   comparacion la vida de los soldado
```

Listing 21: Commits principales

```
commit 741839fff18cd3d15c9dad5557c6fef0c10eec71 (HEAD -> main, origin/main)
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Oct 16 14:29:38 2023 -0500

   Implementa el ordenamiento de vida por insertion

   Se agrega un metodo que ordena a llos soldados por su vida
```

8. Anexo: graphics y soldado

Para lograr representar un soldado en pantalla se uso arte ascii para representar pixeles de distintos colores, en este caso se usaron los simbolos ””para el negro y ”.”para el blanco

- El soldado tiene 58x58 en caracteres
- Se hizo representado con una fuente monoespaciada para no perder proporcion
- Para soldados de diferentes colores se puede invertir los simbolos usados

Listing 22: Representacion en ascii del soldado

```
public static Picture soldier(){
  String [] img = {
        "                                                ",
        "                                                ",
        "                                                ",
        "                                                ",
        "                        #######                 ",
        "                      ##########                ",
        "                     ##.........##              ",
        "                     ##...........##            ",
        "                    ##.............##          ",
        "           ###        ##.........##            ",
        "         #######      ##.........##            ",
        "    ###################  ##.......##            ",
        "       ###################  ##....##            ",
        "           ##.....#####   ##......##            ",
        "             ##....#     ##........###          ",
        "              ##...##   ##............##        ",
        "              ##...##  ##..............##       ",
        "               ##...## ##................##     ",
        "               ##...###..................##     ",
        "                ##......##...............##     ",
        "                 ###..## ##.............##      ",
        "                   ####  ##............##       ",
        "                         ##..............###    ",
        "                        ##..................##  ",
        "                        ##..................##  ",
        "                       ##....................##  ",
        "                      ##......................##  ",
        "                      ##......................##  ",
        "                     ##......................##   ",
        "                      ##..................#####   ",
        "                     ##........##.......##        ",
        "                    ##.........####......##       ",
        "                   ##.........## ##.....##        ",
        "                   ##.........## ##.....##        ",
        "                  ##.........## ##......##        ",
        "                  ##........## ##......##         ",
        "                  ##........##  ##......###      ",
        "                  ##......##     ##.......###     ",
        "                 ##......##       ##.......##     ",
        "                 ##......##        ##.......##    ",
        "                ##......##          ##.......##   ",
        "                 ##....###           ######.....##  ",
        "                  ##..##                ##.....##  ",
        "                  ######                  ###...##  ",
        "                 #######                   #######  ",
        "              ###########               ##########  ",
        "             ############              ###########  ",
        "            ###############            ########     ",
        "                 ###                     ###       ",
        "          ########################################## ",
        "        ###......................................### ",
        "        ###......................................### ",
        "          ########################################## ",
        "                                                ",
        "                                                ",
        "                                                ",
        "                                                "
  };
  return new Picture(img);
 }
```

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	1	
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	
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		17	