Informe de Laboratorio 03

Tema: Arreglos 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
03	Arreglos de Objetos	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 20 Setiembre 2023	Al 28 Setiembre 2023

1. Tarea

- Analice, complete y pruebe el Código de la clase DemoBatalla
- Solucionar la Actividad 4 de la Práctica 1 pero usando arreglo de objetos
- Solucionar la Actividad 5 de la Práctica 1 pero usando arreglo de objetos

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

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JhonatanDczel/fp2-23b.git`
- URL para el laboratorio 02 en el Repositorio GitHub.
- `https://github.com/JhonatanDczel/fp2-23b/tree/main/fase01/lab03`

4. Actividad 1: Completar el codigo de DemoBatalla

4.1. Implementacion del metodo Mostrar Naves

- Necesitamos un metodo que muestre los datos de una flota
- Para eso debera recibir un arreglo de naves

Listing 1: Metodo mostrarNaves

```
public static void mostrarNaves(Nave [] flota){
  for(int i = 0; i < flota.length; i++){
    System.out.println("Nave numero " + i + ":");
    System.out.println(flota[i].getNombre());
  }
}
```

- Como vemos, el metodo recorre el arreglo y va imprimiendo el nombre de cada nave que encuentra en la posicion i
- para evitar un error de desbordamiento de limites se usa flota.length
- Ahora haremos una optimizacion usando el ciclo for each

Listing 2: Optimizacion del metodo mostrarNaves

```
public static void mostrarNaves(Nave [] flota){
  System.out.println("Mostrando las naves creadas: ");
  for(Nave n : flota){
    System.out.println(n.getNombre());
  }
}
```

- Esta version es mas eficiente ya que no es necesario conocer la ubicacion del indice

4.2. Ejecucion

Listing 3: Ejecucion del codigo

```
  Naves creadas:
  Nave numero 1:
  Ninha
  Nave numero 2:
  Pinta
  Nave numero 3:
  SantaMaria
```

- Asi es como se ve la ejecucion en consola, con un ejemplo generico

4.3. Metodo mostrar naves por nombre

Listing 4: Metodo mostrarPorNombre

```java
public static void mostrarPorNombre(Nave [] flota, String nombre){
    System.out.println();
    int i = 1;
    for(Nave n : flota){
      if(n.getNombre().equals(nombre)){
        System.out.println("Nave " + i + ":");
        mostrarNave(n);
        i++;
      }
    }
    if(i == 1)
      System.out.println("No se han encontrado naves con ese nombre");
  }
```

- Como podemos ver, el metodo recorre el arreglo que se le da en busca de naves con el mismo nombre
- Para trabajar sin distinciones entre mayusculas y minusculas implementaremos el metodo toLowerCase, para trabajar estandarizadamente y evitar conflictos:

Listing 5: Metodo mostrarPorNombre

```java
public static void mostrarPorNombre(Nave [] flota, String nombre){
  nombre = nombre.toLowerCase();
  System.out.println();
  int i = 1;
  for(Nave n : flota){
    if(n.getNombre().toLowerCase().equals(nombre)){
      System.out.println("Nave " + i + ":");
      mostrarNave(n);
      i++;
    }
  }
  if(i == 1)
    System.out.println("No se han encontrado naves con ese nombre");

}
```

4.4. Metodo Mostrar por puntos

- Ahora necesitamos un metodo que muestre las naves con cantidad de puntos igual o menor al que el usuario entre por teclado

Listing 6: Implementacion del metodo mostrar por puntos

```java
public static void mostrarPorPuntos(Nave [] flota, int pts){
  System.out.println();
  int i = 1;
  for(Nave n : flota){
    if(n.getPuntos() <= pts){
      System.out.println("Nave " + i + ":");
      mostrarNave(n);
      System.out.println();
      i++;
    }
```

```
  }
}
```

- Recorremos el arreglo comprobando si los puntos son iguales o menores al que se ingreso, en caso serlo se imprimen los datos de la nave
- Ahora necesitamos un caso ”default”por si no se encuentran naves

Listing 7: Condicion de default

```
if(i == 1){
  System.out.println("No se han encontrado naves");
}
```

4.5. Metodo contarLetrasRestantes

- Ahora necesitamos un metodo para mostrar la nave con mas puntos
- Para eso vamos a recorrer el ciclo y hacer una simple comprobacion con un dato pivote

Listing 8: Metodo Mostrar Mayor Puntos

```
public static Nave mostrarMayorPuntos(Nave [] flota){
  Nave mayor = flota[0];
  for(int i = 0; i < flota.length; i++){
    if(flota[i].getPuntos() > mayor.getPuntos())
      mayor = flota[i];
  }
  return mayor;
}
```

- Recorremos el arreglo en busca del mayor puntaje
- Al final, el metodo devuelve la nave con el mayor puntaje, usando el indice que nos sirvio como indicador del mayor puntaje en cada vuelta

4.6. Metodo para desordenar un array de Naves

- Ahora necesitamos desordenar los elementos de un array de naves dado
- Para eso usaremos la clase Random para generar numeros aleatorios
- La estructura constara de un ciclo for que itere sobre cada uno de los elementos y un ciclo while dentro que se asegure de encontrar lugar para cada uno:

Listing 9: Ciclo while

```
  boolean ubicado = false;
  while(!ubicado){
    int numRandom = random.nextInt(flota.length);
    if(nuevaFlota[numRandom] == null){
      nuevaFlota[numRandom] = flota[i];
      ubicado = true;
      System.out.println("Nave " + i + " ahora ubicada en: " + numRandom);
    }
  }
```

- El ciclo while solo terminara cuando se haya encontrado un lugar vacio para el elemento actual
- El numero random se calcula en funcion a los indices del arreglo, este numca podra ser igual a la longitud para evitar errores de desbordamiento de limite

Listing 10: Metodo completo

```java
public static Nave[] desordenar(Nave[] flota){
  Random random = new Random();
  Nave[] nuevaFlota = new Nave[flota.length];

  for(int i = 0; i < flota.length; i++){
    boolean ubicado = false;
    while(!ubicado){
      int numRandom = random.nextInt(flota.length);
      if(nuevaFlota[numRandom] == null){
        nuevaFlota[numRandom] = flota[i];
        ubicado = true;
        System.out.println("Nave " + i + " ahora ubicada en: " + numRandom);
      }
    }
  }
  return nuevaFlota;

}
```

- Ahora agregamos la estructura for para asegurarnos de recorrer todos los elementos, y finalmente retornamos el arreglo de Naves

4.7. Commits importantes

- Este es un registro de los commits mas importantes
- Estan extraidos del registro de commits al hacer git log

Listing 11: Commits

```
commit 689ce4e09e886234c9a67bf2205bf00e943f6e16
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Sun Sep 24 20:40:02 2023 -0500

    Actividad 1: Implementacion del metodo Mostrar Naves

commit 4a2c4c69527f7d5ce9194f4a28ae5eb8d874424d
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Sun Sep 24 23:25:02 2023 -0500

    Actividad 1: Implementacion del metodo Mostrar Naves Por Nombre

commit 42a4a4e1302014164874c6b1f84c058192d0870a
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Sun Sep 24 23:40:26 2023 -0500

    Actividad 1: Implementacion del metodo Mostrar por Puntos

commit 314a4ec7803f41d98d669fa97931009c74d6f6cb
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Sep 25 00:39:00 2023 -0500

    Actividad 1: Implementacion del metodo para mostrar nave con mayor puntaje

```

```
commit bc49db90fdedaade7c05d8c526608e093f0de518
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Sep 25 10:38:18 2023 -0500

    Actividad 1: Terminando el metodo para desordenar un array de objetos aleatoriamente
```

- Cada commit representa un metodo implementado
- Adicionalmente cada metodo esta acompanhado de su optimizacion

4.8. Ejecucion

- Tendremos ahora una demostracion sobre el DemoBatalla:

Listing 12: Ejecucion en la linea de comandos

```
Naves creadas:
Nave 1:
Nombre: Independencia
Estado: true
Puntos: 65

Nave 2:
Nombre: Huascaran
Estado: false
Puntos: 65

Nave 3:
Nombre: Union
Estado: false
Puntos: 100

Desordenando las naves:
Nave 0 ahora ubicada en: 1
Nave 1 ahora ubicada en: 2
Nave 2 ahora ubicada en: 0

Mostrar naves por nombre, ingrese un nombre:
Union

Nave 1:
Nombre: Union
Estado: false
Puntos: 100

Mostrar naves por puntos, ingrese una cantidad:
35

No se han encontrado naves

Nave con mayor número de puntos: Nave@5e2de80c
```

- Como hemos podido ver, todos los metodos funcionan correctamente

5. Actividad 2: Rehacer la actividad 4 del laboratorio 01, pero con array de Objetos

- En aquel codigo, todo se maneja en variables simples

- Unicamente necesitamos crer un array tal que asi Soldado[], y reemplazar las variables por metodos de Soldado

Listing 13: Array de Objetos

```
Soldado[] ejercito = new Soldado[5];

for(int i = 0; i < 5; i++){
  System.out.println("\nIngrese el nombre del soldado numero " + (i + 1) + ":");
  ejercito[i] = new Soldado(sc.next());

  System.out.println("Ingrese el nivel de vida del soldado numero " + (i + 1) + ":");
  ejercito[i].setLife(sc.nextInt());
}
```

- Ahora, para mostrar los datos, mas de lo mismo, solo tenemos que cambiar variables por metodos

Listing 14: Array de Objetos

```
System.out.println("\n=====DATOS DE SOLDADOS=====");
System.out.println("Soldado: " + ejercito[0].getName() + " \nNivel de vida: " +
    ejercito[0].getLife() + "\n");
System.out.println("Soldado: " + ejercito[1].getName() + " \nNivel de vida: " +
    ejercito[1].getLife() + "\n");
System.out.println("Soldado: " + ejercito[2].getName() + " \nNivel de vida: " +
    ejercito[2].getLife() + "\n");
System.out.println("Soldado: " + ejercito[3].getName() + " \nNivel de vida: " +
    ejercito[3].getLife() + "\n");
System.out.println("Soldado: " + ejercito[4].getName() + " \nNivel de vida: " +
    ejercito[4].getLife() + "\n");
```

5.1. Ejecucion

- A continuacion veremos un ejemplo de ejecucion:

```
Ingrese el nombre del soldado numero 1:
Punchinelo
Ingrese el nivel de vida del soldado numero 1:
12

Ingrese el nombre del soldado numero 2:
Roger
Ingrese el nivel de vida del soldado numero 2:
54

Ingrese el nombre del soldado numero 3:
Klibre
Ingrese el nivel de vida del soldado numero 3:
65

=====DATOS DE SOLDADOS=====
Soldado: Punchinelo
Nivel de vida: 12

Soldado: Roger
Nivel de vida: 54

Soldado: Klibre
Nivel de vida: 65
```

6. Actividad 3: Rehacer la actividad 05 del laboratorio 01, pero con array de Objetos

- Ahora es mas simple aun, dado que en ese laboratorio trabajamos con Arreglo de Strings:

```
  String[] army1 = initializeArmy();
  String[] army2 = initializeArmy();

  System.out.println("                ");
  System.out.println("  Welcome to the Battle  ");
  System.out.println("    Simulator Game!    ");
  System.out.println("                ");

  System.out.println("\n***** Prepare for battle! *****");

  displayArmy(army1);
  displayArmy(army2);

  System.out.println(whoWins(army1, army2));
```

- Deberemos sustiruir los String[] por Soldado[]
- Es un trabajo trivial que no cambia en nada la estructura del codigo.. pero aqui esta

```
public static void main(String[] args){
  Soldado[] army1 = initializeArmy();
  Soldado[] army2 = initializeArmy();

  System.out.println("                ");
  System.out.println("  Welcome to the Battle  ");
  System.out.println("    Simulator Game!    ");
  System.out.println("                ");

  System.out.println("\n***** Prepare for battle! *****");

  displayArmy(army1);
  displayArmy(army2);

  System.out.println(whoWins(army1, army2));

}
```

- Adicionalmente tambien tenemos que modificar los metodos que usa este codigo

6.1. Ejecucion

- La ejecucion es exactamente la misma:

```
  Ingrese el nombre del soldado numero 1: Ricardo
  Ingrese el nivel de vida del soldado numero 1: 54

  Ingrese el nombre del soldado numero 2: Potter
  Ingrese el nivel de vida del soldado numero 2: 65

  Ingrese el nombre del soldado numero 3: Hilario
  Ingrese el nivel de vida del soldado numero 3: 65

```

```
=====DATOS DE SOLDADOS=====
Soldado: Ricardo
Nivel de vida: 54

Soldado: Potter
Nivel de vida: 65

Soldado: Hilario
Nivel de vida: 65
```

7. Commits importantes (Actividad 2 y 3)

- Dada la naturaleza trivial del trabajo, todo se hizo en 2 commits

```
commit ac2abb32f6f077e13f0202653e456ccbeacfee2e (HEAD -> main, origin/main)
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Sep 25 10:58:13 2023 -0500

    Actividad 3: hacer la actividad 5 del laboratorio 1 pero con arreglo de objetos

commit ccd2d7c9d210b83aa4903a69d0eb6879a173ce52
Author: JhonatanDczel <jariasq@unsa.edu.pe>
Date:  Mon Sep 25 10:52:43 2023 -0500

    Actividad 2: hacer La actividad 4 del laboratorio 1 con arreglo de objetos
```

8. Rúbricas

8.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

8.2. 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 2: 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 3: 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	