Informe de Laboratorio 05

Tema: ArrayList

Nota

Estudiante	Escuela	Asignatura
Roni Companocca Checco rcompanocca@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Semestre: II Código: 20210558

Laboratorio	Tema	Duración
05	ArrayList	02 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 29 Septiembre 2023	Al 29 Septiembre 2023

1. TAREA

1.1. Objetivos:

- Crear e inicializar ArrayList
- Realizar búsquedas secuencial y binaria en un ArrayList
- Implementar métodos de ordenamiento en ArrayList
- Solucionar problemas

1.2. Competencias a alcanzar:

- Diseña, responsablemente, sistemas, componentes o procesos para satisfacer necesidades dentro de restricciones realistas: económicas, medio ambientales, sociales, políticas, éticas, de salud, de seguridad, manufacturación y sostenibilidad.
- Aplica de forma flexible, técnicas, métodos, principios, normas, estándares y herramientas de ingeniería necesarias para la construcción de software e implementación de sistemas de información.

2. EQUIPOS, MATERIALES Y TEMAS UTILIZADOS

- Sistema Operativo Windows
- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.

3. URL DE REPOSITORIO GITHUB

- URL para el Repositorio GitHub.
- `https://github.com/RONI-COMPANOCCA-CHECCO`
- URL para el laboratorio 05 en el Repositorio GitHub.
- `https://github.com/RONI-COMPANOCCA-CHECCO/FP2-LAB5`

4. ACTIVIDADES

4.1. EJERCICIO RESUELTO

- - Creamos la Clase ColeccionesList con los siguientes métodos que realizan las operaciones que se pueden realizar con el ArrayList:

```java
// RONI COMPANOCCA CHECCO
// CUI: 20210558
// LABORATORIO 05
// FUNDAMENTOS DE PROGRAMACION
import java.util.List;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

public class ColeccionesList{
    static Scanner consola = new Scanner(System.in);
    // implementacion de la interfaz
    List<String> listaCadenas = new ArrayList<String>();

    public void llenarArrayList(){
        // agregamos elementos
        listaCadenas.add("Arequipa");
        listaCadenas.add("Tacna");
        listaCadenas.add("Moquegua");
        listaCadenas.add("Trujillo");
        listaCadenas.add("Lima");
        listaCadenas.add("Chiclayo");
        listaCadenas.add("Cajamarca");
        // permite duplicados
        listaCadenas.add("Cajamarca");
        listaCadenas.add("Cuzco");
        listaCadenas.add("Arequipa");
        Iterator iterador = listaCadenas.iterator();
        // recorremos y mostramos contenido de lista
        System.out.println(("\nARRAYLIST\n"));
```

```
        while(iterador.hasNext()){
            String elemento = (String) iterador.next();
            System.out.println(elemento);
        }
        // muestra el tamao de la lista
        System.out.println("Tamao: "+listaCadenas.size());
        // retorna un objeto segun su posicion
        int indice = listaCadenas.indexOf("Arequipa");
        System.out.println("Elemento: "+listaCadenas.get(indice)+"Posicion:
            "+indice);
        indice = listaCadenas.indexOf("Lima");
        System.out.println("Elemento: "+listaCadenas.get(indice)+"Posicion:
            "+indice);
    }

    public void buscarEnArrayList(){
        //busqueda de un objeto dentro de la coleccion
        System.out.println("Ingrese un valor a Buscar: ");
        String objetoBuscar = consola.next();

        if(listaCadenas.contains(objetoBuscar)){
            System.out.println(objetoBuscar+" Se encuentra en la Lista");
        }else {
            System.out.println(objetoBuscar+" No se encuentra!!");
        }

    }

    public void eliminarEnArrayList(){
        //eliminar un objeto de la coleccion
        System.out.println("Ingrese un valor a Borrar: ");
        String objetoBorrar1 = consola.next();
        int posicion = listaCadenas.indexOf(objetoBorrar1);
        listaCadenas.remove(posicion);

        //eliminar todos los objeto de la coleccion
        listaCadenas.clear();
        if (listaCadenas.isEmpty()){
            System.out.println("La lista se encuentra vacia");
        }
    }
}
```

- Segundo: Creamos la Clase Main, donde llamamos a nuestra clase y las operaciones implementadas:

```
public class ArrayList01 {
    public static void main(String[] args){
        ColeccionesList coleccion01 = new ColeccionesList();
        System.out.println("--------------------");
        System.out.println("COLECCION ARRAYLIST");
        System.out.println("--------------------");
        coleccion01.llenarArrayList();
        coleccion01.buscarEnArrayList();
        coleccion01.eliminarEnArrayList();
    }
```

```
}
```

- Ejecucion

```
--------------------
COLECCION ARRAYLIST
--------------------

ARRAYLIST

Arequipa
Tacna
Moquegua
Trujillo
Lima
Chiclayo
Cajamarca
Cajamarca
Cuzco
Arequipa
 Tamao : 10
Elemento: ArequipaPosicion: 0
Elemento: LimaPosicion: 4
Ingrese un valor a Buscar:
4
4 No se encuentra!!
Ingrese un valor a Borrar:
4
```

4.2. EJERCICIO PROPUESTO

- Cree un Proyecto llamado Laboratorio5
- Usted deberá crear las dos clases Soldado.java y VideoJuego3.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. Pero ahora el tablero debe ser un ArrayList bidimensional.
- 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).
- la clase VideoJuego3.java

```java
// RONI COMPANOCCA CHECCO
// CUI: 20210558
// LABORATORIO 05
// FUNDAMENTOS DE PROGRAMACION
import java.util.*;

public class VideoJuego3 {

   public static void main(String[] args) {

      //DECLARACION DE VARIABLES Y ARREGLOS NECESARIOS
      ArrayList<Soldado> ejercito1 = new ArrayList();
      ArrayList<Soldado> ejercito2 = new ArrayList();
      ArrayList<ArrayList<Soldado>> tablero = new ArrayList();
      int batallon1, batallon2;
      int vidatotal1=0, vidatotal2=0;
      double promedioVida1=0, promedioVida2=0;

      // BUCLE PARA DESIGNAR LA CANTIDAD DE FILAS Y COLUMNAS DEL TABLERO
      for(int i=0; i<10; i++) {
         tablero.add(new ArrayList<Soldado>());
         for(int j=0; j<10; j++) {
            tablero.get(i).add(new Soldado());
         }
      }

      // CREACION DEL NUMERO DE POSICIONES DE CADA EJERCITO
      batallon1 = aleatorio(1,10);
      batallon2 = aleatorio(1,10);

      // INICIALIZAR ARREGLOS
      inicializarArreglo(ejercito1, batallon1);
      inicializarArreglo(ejercito2, batallon2);

      // GENERAR EJERCITOS VALIDOS
      generarEjercitos(ejercito1, ejercito2);

      //  AADIR  LOS EJERCITOS AL TABLERO
      aadirTablero(ejercito1, tablero);
      aadirTablero(ejercito2, tablero);

      //IMPRIMIR EL TABLERO
      imprimirTablero(tablero);

      //IMPRIMIR LOS SOLDADOS DE MAYOR VIDA DE CADA EJERCITO
      System.out.println("Soldado de mayor vida del ejercito 1");
      SoldadoConMayorVida(ejercito1);
      System.out.println("soldado de mayor vida del ejercito 2");
      SoldadoConMayorVida(ejercito2);

      //IMPRIMIR LA VIDA TOTAL Y EL PROMEDIO DEL EJERCITO 1
      System.out.println("\nEJERCITO 1: ");
      for (int i=0; i<ejercito1.size(); i++) {
         vidatotal1+=ejercito1.get(i).getPuntos();
         promedioVida1 = vidatotal1/(ejercito1.size()*1.0);
      }
```

```
        System.out.println("Vida total: "+vidatotal1);
        System.out.println("Promedio de vida: "+promedioVida1);

        //IMPRIMIR LA VIDA TOTAL Y EL PROMEDIO DEL EJERCITO 2
        System.out.println("\nEJERCITO 2: ");
        for (int i=0; i<ejercito2.size(); i++) {
            vidatotal2+=ejercito2.get(i).getPuntos();
            promedioVida2 = vidatotal2/(ejercito2.size()*1.0);
        }
        System.out.println("Vida total: "+vidatotal2);
        System.out.println("Promedio de vida: "+promedioVida2);

        //IMPRIMIR LOS SOLDADOS CREADOS EN EL ORDEN POR DEFECTO
        System.out.println("\nLista ejercito 1:");
        for(int i=0; i<ejercito1.size(); i++) {
            imprimir(ejercito1.get(i));
        }
        System.out.println("\nLista ejercito 2:");
        for(int i=0; i<ejercito2.size(); i++) {
            imprimir(ejercito2.get(i));
        }

        // IMPRIMIR LOS DATOS DE LOS SOLDADOS ORDENADOS DE MAYOR A MENOR DEPENDIENDO
            DE SU NIVEL DE VIDA USANDO DOS TIPOS DE ALGORITMO
        ordenarPorVidaMetodoA(ejercito1);
        ordenarPorVidaMetodoB(ejercito2);
        System.out.println("\nEjercito 1 Ordenados por nivel de vida");
        for(int i=0; i<ejercito1.size(); i++) {
            imprimir(ejercito1.get(i));
        }
        System.out.println("\nEjercito 2 Ordenados por nivel de vida");
        for(int i=0; i<ejercito2.size(); i++) {
            imprimir(ejercito2.get(i));
        }

        // MOSTRAR EJERCITO GANADOR LA METRICA USADA ARA DESIGNAR AL GANADOR ES POR
            EL NIVEL DEL PROMEDIO DE VIDA DE CADA EJERCITO
        if(promedioVida1>promedioVida2) {
            System.out.println("\nGANADOR ***EJERCITO 1***");
        }else if (promedioVida1<promedioVida2) {
            System.out.println("\nGANADOR ***EJERCITO 2***");
            }
        else {
            System.out.print("\n***ES UN EMPATE***");
        }
    }

    // METODO PARA CREAR NUMEROS ALEATORIOS EN UN RANGO
    public static int aleatorio(int min, int max) {
        return(int)(Math.random()*(max-min+1)+min);
    }

    // METODO PARA INICIAR UN ARRAYLIST
    public static void inicializarArreglo (ArrayList<Soldado> soldadito, int num) {
        for (int i=0; i<num; i++) {
            soldadito.add(new Soldado());
```

```
   }
}

// METODO PARA GENERAR DATOS DEL OBJETO SOLDADO
public static Soldado generarDatos() {
   Soldado soldadito = new Soldado();
   soldadito.setPuntos(aleatorio(1,5));
   soldadito.setColumna(aleatorio(1,10));
   soldadito.setFila(aleatorio(1,10));
   return soldadito;
}

// METODOS PARA GENERAR LOS EJERCITOS DE MANERA ALEATORIA
public static void generarEjercitos(ArrayList<Soldado>B1, ArrayList<Soldado>B2) {
   ArrayList<Soldado>Soldados = new ArrayList();
   Soldados.add(generarDatos());
   for (int i=1; i<(B1.size()+B2.size()); i++) {
      Soldados.add(generarDatos());
      for (int j=0; j<i; j++) {
         if(Soldados.get(i).getFila()==Soldados.get(j).getFila()) {
            if(Soldados.get(i).getColumna()==Soldados.get(j).getColumna()){
               Soldados.remove(i);
               i--;
            }
         }
      }
   }
   for (int i=0; i<B1.size(); i++) {
      B1.add(i, Soldados.get(i));
      B1.get(i).setNombre("Soldado"+i+"x1");
      B1.get(i).setColumn(B1.get(i).getPuntos()+"[E1]");
      B1.remove(i+1);
   }
   for (int i=0; i<B2.size(); i++) {
      B2.add(i, Soldados.get(i+B1.size()));
      B2.get(i).setNombre("Soldado"+i+"x2");
      B2.remove(i+1);
      B2.get(i).setColumn(B2.get(i).getPuntos()+"[E2]");
   }
}

// METODO PARA AADIR LOS EJERCITOS AL TABLERO
public static void aadirTablero(ArrayList<Soldado>soldadito,
    ArrayList<ArrayList<Soldado>>table) {
   for (int i=0; i<soldadito.size(); i++) {
      table.get(soldadito.get(i).getColumna()-1).add(soldadito.get(i).getFila()-1,soldadito.get(i));
      table.get(soldadito.get(i).getColumna()-1).remove(soldadito.get(i).getFila());
   }
}

// METODO PARA IMPRIMIR EL TABLERO EN LA CUAL SE DESARROLLA EL JUEGO
public static void imprimirTablero(ArrayList<ArrayList<Soldado>> table) {
   System.out.println("\tA\tB\tC\tD\tF\tG\tH\tI\tJ");
   for(int i=0; i<table.size(); i++) {
      System.out.print(i+1);
      for(int j=0; j<table.get(i).size();j++) {
```

```
            System.out.print("\t"+table.get(i).get(j).getColumn());
         }
         System.out.println("\n");
      }
   }

   //METODO PARA IMPRIMIR LOS SOLDADOS DE MAYOR VIDA
   public static void SoldadoConMayorVida (ArrayList<Soldado>soldadito) {
      Soldado mayor = new Soldado();
      mayor.setPuntos(0);
      for(int i=0; i<soldadito.size();i++) {
         if (mayor.getPuntos()<soldadito.get(i).getPuntos()) {
            mayor = soldadito.get(i);
         }
      }
      imprimir(mayor);
   }

   // METODO PARA IMPRIMIR EL NOMBRE, LA POSICION Y NIVEL DE VIDA DEL SOLDADO
   public static void imprimir(Soldado soldadito) {
      System.out.println("Nombre: "+soldadito.getNombre()+"\nPosicion:
          "+soldadito.getColumna()+"X"+soldadito.getFila()+"\tVida:
          "+soldadito.getPuntos());
   }

   // METODO QUE NOS AYUDA A ORDENAR LOS SOLDADOS DE ACUERDO A SU NIVEL DE VIDA,
       USUANDO UN ALGORITMO DE ORDENAMIENTO DE BURBUJA
   public static void ordenarPorVidaMetodoA(ArrayList<Soldado>soldadito) {
      Soldado aux = new Soldado();
      for(int i=0; i<soldadito.size()-1; i++) {
         for(int j=0; j<soldadito.size()-i-1; j++) {
            if(soldadito.get(j).getPuntos()<soldadito.get(j+1).getPuntos()) {
               aux = soldadito.get(j);
               soldadito.set(j,soldadito.get(j+1));
               soldadito.set(j+1,aux);
            }
         }
      }
   }

    // METODO QUE NOS AYUDA A ORDENAR LOS SOLDADOS DE ACUERDO A SU NIVEL DE VIDA,
        EN ESTA OCACION DIFERENTE A LA ANTERIOR QUE ERA ALGORITMO DE BURBUJA
    public static void ordenarPorVidaMetodoB(ArrayList<Soldado> soldadito) {
        Collections.sort(soldadito, new Comparator<Soldado>() {
            public int compare(Soldado s1, Soldado s2) {
                // Orden descendente por puntos de vida
                return Integer.compare(s2.getPuntos(), s1.getPuntos());
            }
        });
    }
}
```

- la clase Soldado.java

```
// RONI COMPANOCCA CHECCO
// CUI: 20210558
```

```java
// LABORATORIO 05
// FUNDAMENTOS DE PROGRAMACION - laboratorio
// clase Soldado.java
public class Soldado {
   private String nombre;
   private int fila;
   private int columna;
   private int puntos;
   private String column;

   public Soldado() {
      nombre = "";
      fila = 0;
      columna = 0;
      puntos = 0;
      column = "";
   }

   // METODOS MUTADORES
   public void setNombre(String nombre) {
      this.nombre = nombre;
   }

   public void setFila(int fila) {
      this.fila = fila;
   }

   public void setColumna(int columna) {
      this.columna = columna;
   }

   public void setPuntos(int puntos) {
      this.puntos = puntos;
   }

   public void setColumn(String column) {
      this.column = column;
   }

   // METODOS ACCESORES
   public String getNombre() {
      return nombre;
   }

   public int getFila() {
      return fila;
   }

   public int getColumna() {
      return columna;
   }

   public int getPuntos() {
      return puntos;
   }
```

```
   public String getColumn() {
      return column;
   }
}
```

- Ejecucion

```
      A      B      C      D      F      G      H      I      J
1

2                   1[E2]

3

4

5                   3[E2]                 1[E1]

6             2[E2]                2[E2]  2[E2]

7                   5[E2]                 4[E2]

8

9

10

Soldado de mayor vida del ejercito 1
Nombre: Soldado0x1
Posicion: 5X6  Vida: 1
soldado de mayor vida del ejercito 2
Nombre: Soldado4x2
Posicion: 7X3  Vida: 5

EJERCITO 1:
Vida total: 1
Promedio de vida: 1.0

EJERCITO 2:
Vida total: 19
Promedio de vida: 2.7142857142857144

Lista ejercito 1:
Nombre: Soldado0x1
Posicion: 5X6  Vida: 1

Lista ejercito 2:
Nombre: Soldado0x2
Posicion: 6X5  Vida: 2
Nombre: Soldado1x2
Posicion: 2X3  Vida: 1
Nombre: Soldado2x2
Posicion: 6X2  Vida: 2
Nombre: Soldado3x2
Posicion: 6X6  Vida: 2
```

```
Nombre: Soldado4x2
Posicion: 7X3 Vida: 5
Nombre: Soldado5x2
Posicion: 5X3 Vida: 3
Nombre: Soldado6x2
Posicion: 7X6 Vida: 4

Ejercito 1 Ordenados por nivel de vida
Nombre: Soldado0x1
Posicion: 5X6 Vida: 1

Ejercito 2 Ordenados por nivel de vida
Nombre: Soldado4x2
Posicion: 7X3 Vida: 5
Nombre: Soldado6x2
Posicion: 7X6 Vida: 4
Nombre: Soldado5x2
Posicion: 5X3 Vida: 3
Nombre: Soldado0x2
Posicion: 6X5 Vida: 2
Nombre: Soldado2x2
Posicion: 6X2 Vida: 2
Nombre: Soldado3x2
Posicion: 6X6 Vida: 2
Nombre: Soldado1x2
Posicion: 2X3 Vida: 1

GANADOR ***EJERCITO 2***
```

5. CUESTIONARIO

5.1. ¿Cómo se declara e inicializa un ArrayList bidimensional?

- En Java, no existe un tipo de dato ArrayList bidimensional como tal. Sin embargo, puedes lograr una funcionalidad similar utilizando un ArrayList de ArrayLists, lo que te permitirá crear una estructura de datos bidimensional.

```
ArrayList<ArrayList<Integer>> matriz = new ArrayList<>();
```

5.2. ¿Qué ventajas tienen los ArrayList con respecto a los arreglos?

- Los ArrayList en Java tienen varias ventajas con respecto a los arreglos (arrays) que los hacen una elección más flexible y poderosa en muchas situaciones: Tamaño dinámico, Inserción y eliminación eficiente, Flexibilidad, Métodos útiles, Uso de autoboxing, etc.

5.3. ¿Mencione 4 métodos importantes del ArrayList? ¿Qué finalidad tienen?

- add(E elemento): Este método se utiliza para agregar un elemento al final del ArrayList. La finalidad principal de este método es aumentar la capacidad del ArrayList si es necesario y agregar el elemento especificado al final de la lista.

- remove(int índice): Este método se usa para eliminar el elemento en el índice especificado del ArrayList. La finalidad principal es eliminar un elemento en una posición particular de la lista.
- get(int índice): Este método se emplea para obtener el elemento en el índice especificado del ArrayList. La finalidad es recuperar un elemento en una posición particular de la lista.
- size(): Este método devuelve el número de elementos en el ArrayList. La finalidad es conocer la cantidad de elementos presentes en la lista.

6. REFERENCIAS

- M. Aedo, "Fundamentos de Programación 2 - Tópicos de Programación Orientada a Objetos", Primera Edición, 2021, Editorial UNSA.
- `https://github.com/rescobedoq/programacion.git`
- J. Dean, Ïntroduction to programming with Java: A Problem Solving Approach", Third Edition, 2021, McGraw-Hill.
- C. T. Wu, .ªn Introduction to Object-Oriented Programming with Java", Fifth Edition, 2010, McGraw-Hill.
- P. Deitel, "Java How to Program", Eleventh Edition, 2017, Prentice Hall.