

Informe de Laboratorio 03

Tema: Clases Genéricas

Nota

Estudiante	Escuela	Asignatura
Misael Marrón Lope mmarronl@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	EDA Semestre: III Código: 20220575

Laboratorio	Tema	Duración
03	Clases Genéricas	

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	2023	5 junio 2023

1. Tarea

- Implementa una Lista usando POO con tipos genéricos siguiendo los estándares de Java. (Los métodos para una lista).
- Implemente una clase Node donde T es un tipo genérico, esta clase debe contener al menos dos propiedades.
- Utilizar Git para evidenciar su trabajo.
- Enviar trabajo al profesor en un repositorio GitHub Privado, dándole permisos como colaborador.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 10 ver. 22H2
- Eclipse IDE, Visual studio
- java 20.0.1
- Git 2.40.1.
- Cuenta en GitHub con el correo institucional.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/MisaelMarron/eda-lab-b-23a.git>
- URL para el laboratorio 02 en el Repositorio GitHub.
- <https://github.com/MisaelMarron/eda-lab-b-23a/tree/main/lab02>

4. Actividades : La creacion de List y Nodo

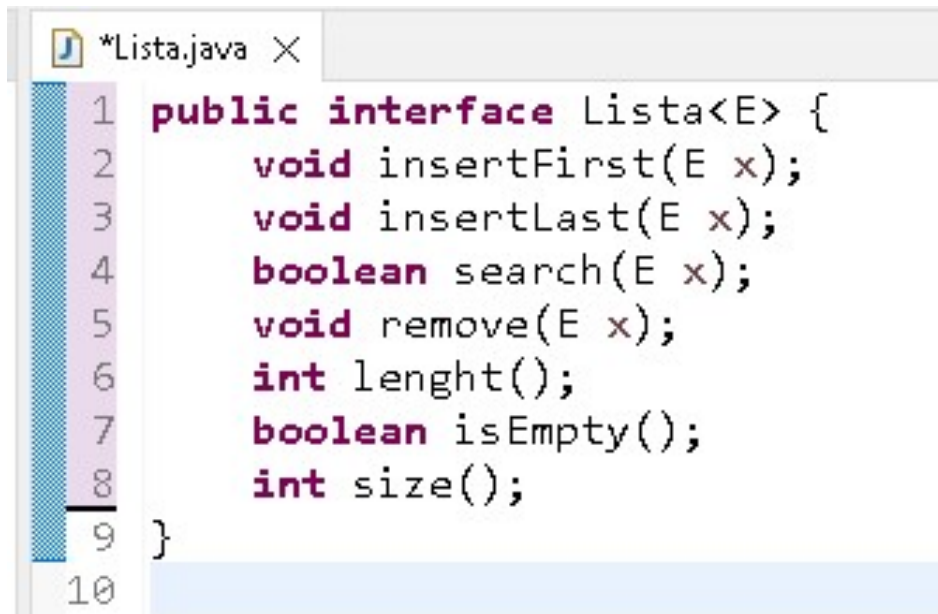
- Tendremos que crear clases genericas con List usando metodos que se indican en el material proporcionado por el ingeniero Richard.

4.1. Commits Importantes:

Listing 1: Mi primer commit importante es cuando cree la clase lista y le agregue sus metodos de interface , para ser usados luego.

```
commit 1c27525bc83afc8eb2c28e2aaa978dd930a43ad6
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date: Tue May 30 11:47:10 2023 -0500
```

Agregamos la clase lista con sus metodos



```
*Lista.java X
1 public interface Lista<E> {
2     void insertFirst(E x);
3     void insertLast(E x);
4     boolean search(E x);
5     void remove(E x);
6     int lenght();
7     boolean isEmpty();
8     int size();
9 }
10
```

Listing 2: Mi segundo commit importante es cuando cambie la clase lista y agregue la clase nodo.

```
commit 024e48665cbea91cd1e0153dee2db4d0d92ab6b1
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date: Tue May 30 11:53:27 2023 -0500
```

agregamos la clase Nodo y cambiamos metodos

```
1 public interface Lista<E> {
2     void add(E x);
3     boolean contains(E x);
4     void remove(E x);
5     boolean isEmpty();
6     int size();
7 }
8
```

```
*Lista.java  Nodo.java X
1 public class Nodo<E> {
2     private E data;
3     private Nodo<E> next;
4
5     public Nodo(E data, Nodo<E> next) {
6         this.data = data;
7         this.next = next;
8     }
9     public Nodo(E data) {
10        this(data, null);
11    }
12    public E getData() {
13        return this.data;
14    }
15    public void setData(E data) {
16        this.data = data;
17    }
18    public Nodo<E> getNext() {
19        return this.next;
20    }
21    public void setNext(Nodo<E> next) {
22        this.next = next;
23    }
24    public String toString() {
25        return this.data.toString();
26    }
27 }
```

Listing 3: Mi tercer commit mas importante es cuando agregue ListaUsar y complete con metodos.

commit 8bf9e781c0d0140be15c5d194b5e0b68a5a89096
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date: Tue May 30 11:59:05 2023 -0500

Agregamos los metodos necesarios

```

1 public class ListaUsar<E> implements Lista<E>{
2     private Nodo<E> root;
3     private int count;
4
5     public ListaUsar() {
6         this.root = null;
7         this.count = 0;
8     }
9
10    public boolean isEmpty() {
11        return this.root == null;
12    }
13
14    public int size() {
15        return this.count;
16    }
17
18    public void insertFirst(E x) {
19        this.root = new Nodo<E>(x, this.root);
20        this.count ++;
21    }
22
23    public void add(E x) {
24        if (isEmpty())
25            insertFirst(x);
26        else {
27            Nodo<E> aux = this.root;
28            while (aux.nextNodo() != null)
29                aux = aux.nextNodo();
30            aux.setNext(new Nodo<E>(x));
31            this.count ++;
32        }
33    }
34
35    public boolean contains(E x) {
36        Nodo<E> aux = this.root;
37        for(; aux != null && !aux.getData().equals(x); aux = aux.nextNodo());
38        return aux != null;
39    }
40
41    public void remove(E x) {
42        if (!isEmpty()) {
43            if (this.root.getData().equals(x)) {
44                this.root = this.root.nextNodo();
45                this.count --;
46            }
47            else {
48                Nodo<E> aux = this.root;
49                while(aux.nextNodo() != null && !aux.nextNodo().getData().equals(x))
50                    aux = aux.nextNodo();
51                if (aux.nextNodo() != null) {
52                    aux.setNext(aux.nextNodo().nextNodo());
53                    this.count --;
54                }
55            }
56        }
57    }
58
59    public String toString() {
60        String str = "";
61        for(Nodo<E> aux = this.root; aux != null; aux = aux.nextNodo())
62            str += aux.toString() + " ";
63    }

```

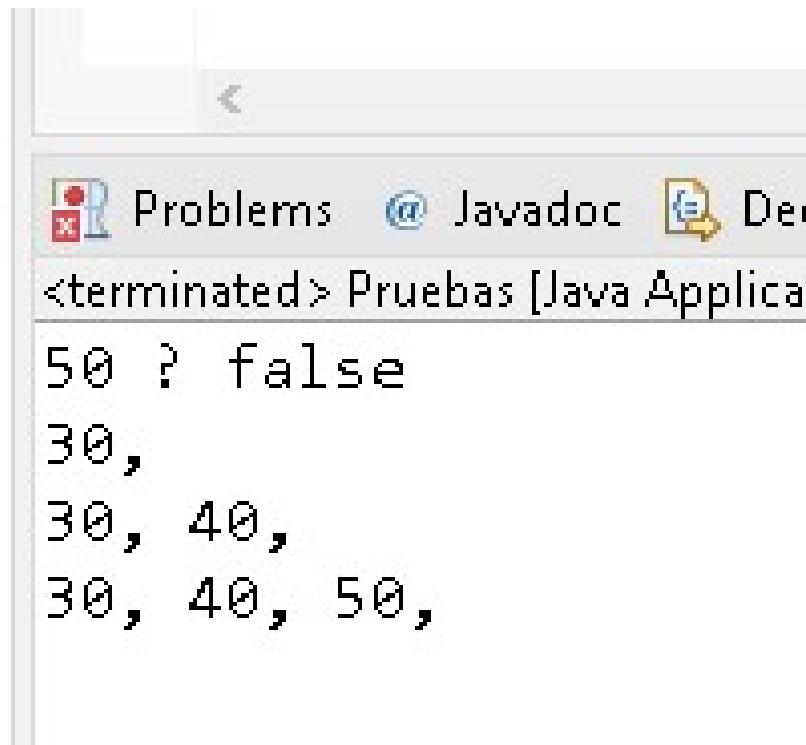
Listing 4: Mi cuarto commit es cuando termine todo el lab03 y agregue una ejecucion en una clase llamada Pruebas.

```
commit a5ec5364040161df3d30695f143e38e2e6cd1a6c
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date: Tue May 30 12:06:23 2023 -0500
```

Agregamos Pruebas para hacer pruebas y terminamos

```
Listajava  Nodojava  ListaUsarjava  Pruebas.java X
1 public class Pruebas {
2     public static void main(String[] args) {
3         Lista<Integer> lista = new ListaUsar<Integer>();
4         System.out.println("50 ? " + lista.contains(50));
5         lista.add(30);
6         System.out.println(lista);
7         lista.add(40);
8         System.out.println(lista);
9         lista.add(50);
10        System.out.println(lista);
11
12    }
13 }
```

- Ejecución en consola del lab03:



```
<terminated> Pruebas [Java Applica
50 ? false
30,
30, 40,
30, 40, 50,
```

4.2. Estructura de laboratorio 03

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab01/  
|--- codigo  
| |--- Lista.java  
| |--- ListaUsar.java  
| |--- Nodo.java  
| |--- Pruebas.java  
|--- latex  
| |--- img  
| | |--- logo_abet.png  
| | |--- logo_episunsa.png  
| | |--- logo_unsa.jpg  
| | |--- codigo1.jpg  
| | |--- codigo2.jpg  
| | |--- codigo2b.jpg  
| | |--- codigo3.jpg  
| | |--- codigo4.jpg  
| | |--- commit01.jpg  
|--- Lab03-MisaelMarron.pdf  
|--- Lab03-MisaelMarron.tex
```

5. Preguntas:

- En este caso no se dejaron preguntas.

6. Rúbricas

6.1. 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.

6.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

Puntos	Nivel			
	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	3	
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	1.5	
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 están 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	4	
Total		20		17	

7. Referencias

- <https://docs.oracle.com/javase/tutorial/java/generics/types.html>
- <https://www.eclipse.org/downloads/packages/release/2022-03/r/eclipse-ide-enterprise-java-and-webtools>
- <https://www.w3schools.com/java/default.asp>