	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): JOSE ANTONIO AYALA BARBOSA

Asignatura: Programación Orientada a Objetos

Grupo: 02

No de Práctica(s): 03 Utilerías y clase de uso general.

Integrante(s): Hernández Reyes Rebeca Sarai

No. de lista o brigada:

Semestre: 2026-1

Fecha de entrega: 11 – 09 – 2025

Observaciones:

CALIFICACIÓN: _____

PRÁCTICA 3 Utilerías y clase de uso general.

PROGRAMACIÓN ORIENTADA A OBJETOS

Hernández Reyes Rebeca Sarai

OBJETIVO

Utilizar bibliotecas propias del lenguaje para realizar algunas tareas comunes y recurrentes.

INTRODUCCIÓN

En esta práctica se busca reforzar el uso de las bibliotecas propias del lenguaje de programación, con el fin de comprender cómo estas facilitan la resolución de tareas comunes y recurrentes. Para ello, se explorará el API del lenguaje, identificando las bibliotecas de uso más frecuente y aplicándolas en la implementación de colecciones que permitan gestionar conjuntos de datos de manera eficiente. Con estas actividades se pretende que el estudiante desarrolle habilidades prácticas para aprovechar los recursos ya disponibles en el lenguaje, sentando así una base sólida para la construcción de programas más robustos y escalables en el futuro.

ACTIVIDADES

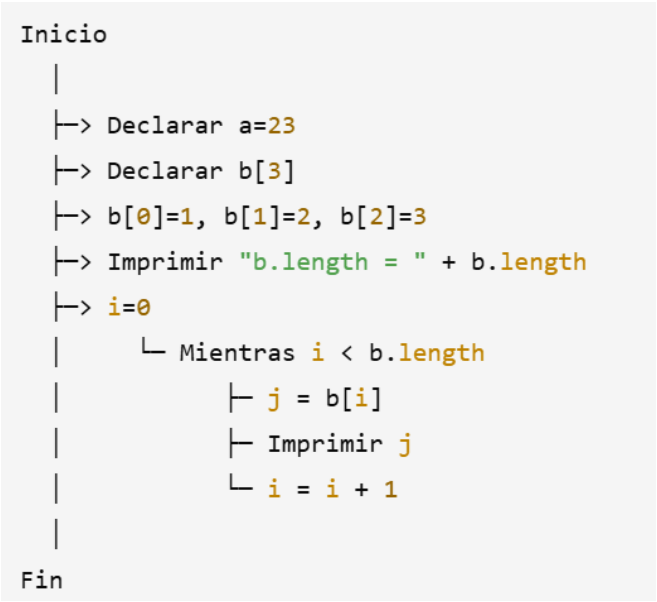
- Conocer el API del lenguaje.
- Reconocer las bibliotecas de uso común.
- Implementar colecciones.

CÓDIGO FUENTE Y DESARROLLO

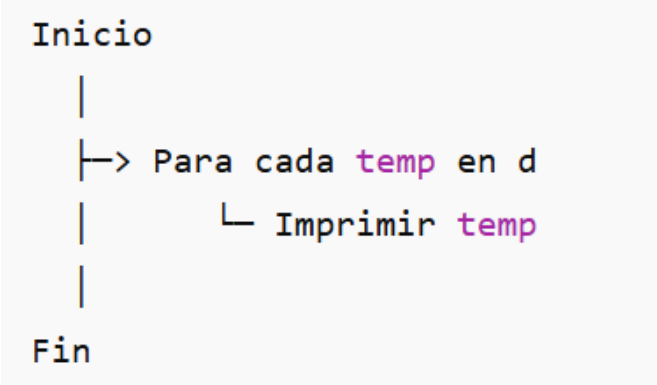
Se desarrolló un programa en el que se declararon arreglos y variables de diferentes tipos de datos, mostrando su tamaño y recorriéndolos mediante ciclos for y for-each. Con esto se reforzó el uso de estructuras iterativas y la forma en que los datos se organizan y manipulan en memoria.

```
System.out.println("###############ARREGLO############");
int a = 23;
int b[]= new int [3];
b[0]=1;
b[1]=2;
b[2]=3;

System.out.println("b.length = " + b.length);

for (int i = 0; i < b.length; i++) {
    int j = b[i];
    System.out.println(j);

}
int[] c = {1,2,3,4,5};
int[] d;
d = new int [10];
```

[Imagen 1. Arreglos.]



[Imagen 2. Diagrama de Flujo arreglo b.]



[Imagen 3. Diagrama de Flujo arreglo c y d.]

```java
System.out.println("############FOR############");
for (int i = 0; i < 10; i++) {
    System.out.println("CONTANDO HACIA ARRIBA:" + i);
}
for (int i = 0; i < d.length; i++) {
    d[i]= i*100;
    System.out.println(i);

}
System.out.println("########FOR EACH##########");
for(int temp:d){
    System.out.println(temp);
}
```

[Imagen 4. For y For Each.]

Posteriormente, se trabajó con cadenas de caracteres, utilizando el operador punto para acceder a sus métodos, como obtener la longitud o concatenar valores. También se incorporaron ejemplos con clases envolventes (wrappers) para transformar tipos primitivos en objetos y viceversa.

```java
System.out.println("######CADENA DE CARACTERES######");
String s = new String("Hola Mundo");
System.out.println(s);
String s1 = "Hola Mundo en s1";
System.out.println(s1);

String nombre = "Rebeca ";
String apellido = "Hernandez";
String nombreCompleto = nombre + apellido;
System.out.println(nombreCompleto);
```

[Imagen 5. Cadena.]

```
Inicio
  |
  ├-> s = "Hola Mundo"
  ├-> Imprimir s
  ├-> s1 = "Hola Mundo en s1"
  ├-> Imprimir s1
  ├-> nombre = "Rebeca "
  ├-> apellido = "Hernandez"
  ├-> nombreCompleto = nombre + apellido
  ├-> Imprimir nombreCompleto
  |
Fin
```

[Imagen 6. Diagrama de cadenas]

```java
System.out.println("#######OPERADOR PUNTO#######");
System.out.println("Arreglo con " + d.length + " elementos");
System.out.println("s tiene " + nombreCompleto.length() + " caracteres");
char[] hola2 ={'h', 'o', 'l', 'a'};
System.out.println("Elementos en hola = " + hola2.length);
//String holaCadena = hola2.toString();
//System.out.println(holaCadena);
```

[Imagen 7. Operador punto.]

```
Inicio
  |
  ├-> Imprimir "Arreglo con " + d.length + " elementos"
  ├-> Imprimir "s tiene " + nombreCompleto.length() + " caracteres"
  ├-> hola2 = {'h','o','l','a'}
  ├-> Imprimir "Elementos en hola = " + hola2.length
  |
Fin
```

[Imagen 8. Diagrama de Operador punto]

```java
System.out.println("#######WRAPPERS#######");
int f = 35;
Integer g = new Integer(35);
Integer h = 35;
String s3 = h.toString();
System.out.println(s3);
```

[Imagen 9. Wrappers.]

```
Inicio
  |
  ├-> f = 35
  ├-> g = new Integer(35)
  ├-> h = 35     // autoboxing
  ├-> s3 = h.toString()
  ├-> Imprimir s3
  |
Fin
```

[Imagen 10. Diagrama de Wrappers]

Asimismo, se implementaron colecciones dinámicas con ArrayList, donde se exploró la inserción de elementos en distintas posiciones y su recorrido con ciclos. Se incluyó el uso de Hashtable para almacenar pares clave–valor, mostrando los datos mediante iteraciones con values() y con una enumeración (Enumeration) para recorrer las llaves.

```java
ArrayList<Integer> miArrayList = new ArrayList<Integer>();
miArrayList.add(4);
miArrayList.add(11);
System.out.println(miArrayList.size());
System.out.println(miArrayList.get(1));
miArrayList.add(0,22);
System.out.println(miArrayList.get(0));
System.out.println("####");
for (Integer integer : miArrayList) {
    System.out.println(integer);

}
miArrayList.add(2,8);
for (Integer integer : miArrayList) {
    System.out.println(integer);

}
```

[Imagen 11. ArrayList.]

```
Inicio
  |
  ├─> Crear miArrayList<Integer>
  ├─> add(4), add(11)
  ├─> Imprimir size()
  ├─> Imprimir get(1)
  ├─> add(0,22)
  ├─> Imprimir get(0)
  ├─> Para cada integer en miArrayList
  |       └─ Imprimir integer
  ├─> add(2,8)
  ├─> Para cada integer en miArrayList
  |       └─ Imprimir integer
  |
Fin
```

[Imagen 12. Diagrama de ArrayList]

```java
System.out.println("######HASHTABLE######");
//<key,value>
//key -> Set
//value -> List

Hashtable<Integer,String> miTabla = new Hashtable<Integer,String>();
miTabla.put(31934567, "Antonio Ayala");
miTabla.put(31229900,"Claudia Jimenez");
System.out.println("Elementos en la tabla = " + miTabla.size());
miTabla.put(31225577, "Alejandro Perez");
System.out.println("Elementos en la tabla = " + miTabla.size());
for (String valor : miTabla.values()) {
    System.out.println(valor);

}
```

[Imagen 13. HashTable.]

```
Inicio
  |
  ├─> Crear miTabla<Integer,String>
  ├─> put(31934567,"Antonio Ayala")
  ├─> put(31229900,"Claudia Jimenez")
  ├─> Imprimir "Elementos = " + size()
  ├─> put(31225577,"Alejandro Perez")
  ├─> Imprimir "Elementos = " + size()
  ├─> Para cada valor en miTabla.values()
  |       └─ Imprimir valor
  |
Fin
```

[Imagen 14. Diagrama de Hashtable]

```java
System.out.println("######ENUMERACION#######");
Integer llave;
String valor;

Enumeration<Integer> llaves = miTabla.keys();
while(llaves.hasMoreElements()){
    llave = llaves.nextElement();
    valor = miTabla.get(llave);
    System.out.println("Elemento < " + llave + " , " + valor + " >");
}
```

[Imagen 15. Enumeración.]

```
Inicio
  |
  ├─> llaves = miTabla.keys()
  ├─> Mientras llaves.hasMoreElements()
  |       ├─ llave = llaves.nextElement()
  |       ├─ valor = miTabla.get(llave)
  |       └─ Imprimir "Elemento < llave , valor >"
  |
Fin
```

[Imagen 16. Diagrama de Enumeración]

Finalmente, se utilizaron bibliotecas comunes del lenguaje como Math, aplicando funciones matemáticas predefinidas (valor absoluto, raíz cuadrada, potencias, etc.), así como Date y Calendar para obtener y mostrar la fecha actual. Todo esto permitió reconocer y aplicar el uso de bibliotecas estándar que facilitan la realización de tareas frecuentes en Java.

```
System.out.println("#########MATH########");
System.out.println(Math.PI);
System.out.println(Math.abs(-66));
System.out.println(Math.sqrt(9));
System.out.println(Math.pow(3, 2));
```

[Imagen 17. Math]

```
Inicio
  |
  ├-> Imprimir Math.PI
  ├-> Imprimir Math.abs(-66)
  ├-> Imprimir Math.sqrt(9)
  └-> Imprimir Math.pow(3, 2)
  |
Fin
```

[Imagen 18. Diagrama de Math]

```
System.out.println("Date");
Date hoy = new Date();
System.out.println(hoy);
```

[Imagen 19. Date]

```
Inicio
  |
  ├-> hoy = new Date()
  └-> Imprimir hoy
  |
Fin
```

[Imagen 20. Diagrama de Date]

```
System.out.println("Calendar");
Calendar calendarioHoy = Calendar.getInstance();
System.out.println(calendarioHoy);

String fechaActual = calendarioHoy.get(Calendar.DAY_OF_MONTH) + "de" + (calendarioHoy.get(Calendar.MONTH)+1) + "de" + calendarioHoy
System.out.println(fechaActual);
}
```

[Imagen 21. Calendar]

```
Inicio
  |
  ├-> calendarioHoy = Calendar.getInstance()
  ├-> Imprimir calendarioHoy
  ├-> dia = get(DAY_OF_MONTH)
  ├-> mes = get(MONTH) + 1
  ├-> anio = get(YEAR)
  ├-> fechaActual = dia + "de" + mes + "de" + anio
  └-> Imprimir fechaActual
  |
Fin
```

[Imagen 22. Diagrama de Calendar]

CONCLUSIONES

Durante esta práctica se logró afianzar el uso de arreglos, cadenas de caracteres y colecciones dinámicas, reconociendo sus diferencias y aplicaciones. Además, se comprendió el papel de las clases envolventes y el acceso a métodos de utilidad mediante el operador punto. El trabajo con ArrayList, Hashtable, enumeraciones y bibliotecas estándar como Math, Date y Calendar permitió evidenciar cómo Java provee herramientas listas para usar que simplifican la programación de tareas comunes y recurrentes.

REFERENCIAS

- Thoth, & Thoth. (2023, 4 diciembre). *¿Qué son las APIs de Java?* Formatalent Business School. https://formatalent.com/que-son-las-apis-de-java/
- Amor, R. V. (2020, 6 marzo). *Introducción a Colecciones en Java*. Adictos Al Trabajo. https://adictosaltrabajo.com/2015/09/25/introduccion-a-colecciones-en-java/
- Navas, A. (2023, 4 diciembre). *Clases Wrapper (envoltorio) en Java*. Profile Software Services. https://profile.es/blog/clases-wrapper-envoltorio-en-java/