	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): Calvillo Pérez Juan Ángel

Asignatura: Programación Orientada a Objetos

Grupo: 09

No de Práctica(s): 03

Integrante(s): Durán Rendón Santiago

No. de lista o brigada: 07

Semestre: 2026-1

Fecha de entrega: 11 de septiembre de 2025

Observaciones:

CALIFICACIÓN: _____

Índice

Objetivo 1

Introducción 1

Desarrollo 2

Resultados 7

Conclusión 11

Referencias 11

Recurso adicional 11

Objetivo

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

Introducción

En esta actividad se han reforzado las ideas de tipos de datos, expresiones y control de flujo de programas en Java mediante un menú de programa. Para estructurar la lógica del programa, se ha hecho uso de las estructuras switch y do-while, las cuales permiten ejecutar cada una de las opciones distintas que ofrece el menú hasta que el usuario quiera abandonar el programa. Entre estas opciones, cabe la pena también incluir el uso de arreglos, el tratamiento de cadenas mediante String y StringBuilder, así como el uso de colecciones dinámicas (arrays) como ArrayList y Hashtable, que facilitan el almacenamiento y la gestión de información. También se ha mostrado el uso de la biblioteca de la biblioteca estándar de Java para implementar tareas frecuentes en los programas de software. Con la clase Math se han llevado a cabo tareas de operaciones matemáticas básicas; con Date y Calendar hemos implementado la fecha actual y valores de tiempo más o menos ajustados. Todas estas tareas han permitido entrelazar algunos de los elementos del propio lenguaje cuya asimilación se ha ido reforzando a medida que se ha

implementado el procesamiento del control de flujo y la catalogación de estructuras de datos, formando como consecuencia las bases prácticas de otros programas más complejos que puedan servirse de la noción de un paradigma orientado a objetos.

Desarrollo

```
package javaapplication1.p3;

import java.util.Scanner;
import java.util.Date;
import java.util.Calendar;
import java.util.List;
import java.util.ArrayList;
import java.util.Hashtable;
```

1. Menú con do-while y switch:
    a. Se implementa un ciclo do-while que muestra un menú de opciones al usuario.
    b. El menú permite elegir entre diferentes ejemplos.
    c. El switch controla la lógica para ejecutar el método correspondiente según la opción seleccionada.

```java
public class salon {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int opcion;

        do {
            System.out.println("\n===== MENÙ DE PRÀCTICAS =====");
            System.out.println("1. Arreglos");
            System.out.println("2. Cadenas");
            System.out.println("3. Colecciones");
            System.out.println("4. ArrayList");
            System.out.println("5. Hashtable");
            System.out.println("6. Math, Date y Calendar");
            System.out.println("Elige una opciòn");
            opcion = sc.nextInt();
            sc.nextLine(); //limpiar buffer

            switch (opcion) {
                case 1:
                    ejemploArreglos();
                    break;
                case 2:
                    ejemploCadenas();
                    break;
                case 3:
                    ejemploColecciones();
                    break;
                case 4:
                    ejemploArrayList();
                    break;
                case 5:
                    ejemploHashtable();
                    break;
                case 6:
                    ejemploMathFecha();
                    break;
                case 0:
                    System.out.println("Saliendo del programa...");
                    break;
                default:
                    System.out.println("Opciòn no vàlida");
            }
        } while (opcion != 0);
    }
```

2. Opción 1 – Arreglos:
    a. Se declara un arreglo de enteros con valores iniciales.
    b. Se imprime el primer elemento del arreglo.

c. Se recorre el arreglo con un ciclo for para mostrar todos los elementos.

```
// 1. Arreglos
public static void ejemploArreglos(){
    int[] numeros = {10, 20, 30, 40};
    System.out.println("Primer elemento: " + numeros[0]);
    for(int i = 0; i < numeros.length; i++) {
        System.out.println("Elemento " + i + ":" + numeros[i]);
    }
}
```

3. Opción 2 – cadenas:
    a. Se manipula una cadena para obtener su longitud y mostrarla en mayúsculas.
    b. Se realiza concatenación de cadenas de forma tradicional con +.
    c. Se utiliza StringBuilder para concatenar texto dinámicamente.
    d. Se ejemplifica el uso de la clase Wrapper Integer, convirtiendo una cadena numérica a número entero.

```
//2. Cadenas
public static void ejemploCadenas() {
    String texto = "hola mundo";
    System.out.println("Longitud: " + texto.length());
    System.out.println("Mayùsculas: " + texto.toUpperCase());

    String saludo = texto + "desde Java";
    System.out.println("Concatenado " + saludo);

    StringBuilder sb = new StringBuilder("Hola");
    sb.append(" ").append("Mundo");
    System.out.println("Con append: " + sb);

    Integer numero = Integer.valueOf("123");
    System.out.println("Wrapper Interger: " + (numero + 10));
}
```

4. Opción 3 – Colecciones (List y ArrayList):
    a. Se crea una lista de frutas usando ArrayList.
    b. Se agregan elementos a la lista y se accede a ellos mediante índice.
    c. Se busca la posición de un elemento específico (Mango).
    d. Se elimina un elemento (Pera) y se muestra la lista final con un ciclo for-each.

```java
//3. Colecicciones
public static void ejemploColecciones() {
    List<String> frutas = new ArrayList <>();
    frutas.add("Manzana");
    frutas.add("Pera");
    frutas.add("Mango");

    System.out.println("Primer fruta: " + frutas.get(0));
    System.out.println("Posiciòn de mango: " + frutas.indexOf("Mango"));

    frutas.remove("Pera");

    System.out.println("Lista de frutas:");
    for (String fruta : frutas) {
        System.out.println(fruta);
    }
}
```

5. Opción 4 – ArrayList de colores:
    a. Se construye un ArrayList con valores de colores.
    b. Se accede a un elemento por índice y se imprime el tamaño de la lista.
    c. Se elimina un elemento y se busca la posición de otro.
    d. Se limpia la lista y se verifica si quedó vacía.

```java
//4. ArrayList
public static void ejemploArrayList() {
    ArrayList<String> lista = new ArrayList<>();
    lista.add("Rojo");
    lista.add("Verde");
    lista.add("Azul");

    System.out.println("Elemento en 1: " + lista.get(1));
    System.out.println("Tamaño: " + lista.size());

    lista.remove("Verde");
    System.out.println("Index de Azul: " + lista.indexOf("Azul"));

    lista.clear();
    System.out.println("¿Lista vacia?" + lista.isEmpty());
}
```

6. Opción 5 – Hashtable:
    a. Se crea un Hastable con llaves de tipo Integer y valores String.
    b. Se agregan pares clave-valor.
    c. Se obtiene el valor de una clave específica.

d. Se recorre la tabla con un ciclo for-each mostrando todas las entradas.

```
// 5. Hashtable
public static void ejemploHashtable() {
    Hashtable<Integer, String> tabla = new Hashtable<>();
    tabla.put(1,"Uno");
    tabla.put(2,"Dos");
    tabla.put(3,"Tres");

    System.out.println("Clave 2: " + tabla.get(2));

    for (Integer clave : tabla.keySet()) {
        System.out.println(clave + " -> " + tabla.get(clave));
    }
}
```

7. Opción 6 – Clases de librería estándar (Math, Date, Calendar):
    a. Se emplea Math para operaciones como potencias o raíces cuadradas.
    b. Con Date se muestra la fecha y la hora actuales del sistema.
    c. Con Calendar se define una fecha personalizada y se imprime su valor.

```
//6. Math, Date y Calendar
public static void ejemploMathFecha() {
    System.out.println("2^3 = " + Math.pow(2,3));
    System.out.println("Raìz de 16 = " + Math.sqrt(16));

    Date fecha = new Date();
    System.out.println("Fecha actual: " + fecha);

    Calendar cal = Calendar.getInstance();
    cal.set(2025, Calendar.SEPTEMBER, 4);
    System.out.println("Fecha personalizada: " + cal.getTime());
}
```

8. Opción 0 – Salir:
    a. Se imprime un mensaje y finaliza la ejecución del programa.

```
case 0:
    System.out.println("Saliendo del programa...");
    break;
```

```
===== MEN DE PR CTICAS =====
1. Arreglos
2. Cadenas
3. Colecciones
4. ArrayList
5. Hashtable
6. Math, Date y Calendar
Elige una opci n
```

Resultados

```java
package javaapplication1.p3;

import java.util.*;

public class resultadosp3 {
    public static void main(String[] args) {
        arreglos();
        cadenas();
        colecciones();
        arrayList();
        hashTable();
        mate();
    }

    static void titulo(String t) { System.out.println("\n=== " + t + " ==="); }
```

1. Arreglos:
    a. Hacer un arreglo de reprobados (los de la mesa).
    b. Añadir a una persona del salón.
    c. Quitarme a mí.

```java
    // 1. Arreglos
    static void arreglos() {
        titulo("1) Arreglos");
        String[] reprobados = {"Santiago", "Josue", "Moises", "Mafer"};
        System.out.println("Inicial: " + Arrays.toString(reprobados));

        //Añadir a Ari
        String[] reprobadosMas = Arrays.copyOf(reprobados, reprobados.length + 1);
        reprobadosMas[reprobadosMas.length - 1] = "Ari";
        System.out.println("Añadiendo Ari: " + Arrays.toString(reprobadosMas));

        // Quitarme
        List<String> lista = new ArrayList<>(Arrays.asList(reprobadosMas));
        lista.remove("Santiago");
        System.out.println("Quitando Santiago: " + lista);
    }
```

2. Cadenas:
    a. Un ejemplo de cada uno:
        i. compareTo.
        ii. lastindexOf.
        iii. Endswith.
        iv. Substring.

```
// 2. Cadenas
static void cadenas() {
    titulo("2) Cadenas");
    //compareTo
    String a = "hola", b = "holA";
    System.out.println("compareTo(\"" + a + "\", \"" + b + "\") = " + a.compareTo(b));

    //lastIndexOf
    String texto = "programacion orientada a objetos";
    System.out.println("lastIndexOf('o') = " + texto.lastIndexOf('o'));

    //endsWith
    System.out.println("endsWith(\"tos\") = " + texto.endsWith("tos"));

    //substring
    System.out.println("substring(0, 11) = \"" + texto.substring(0, 11) + "\"");
}
```

3. Colecciones:
    a. HashSet.
    b. TreeSet (duplicados).

```
// 3. Colecciones
static void colecciones() {
    titulo("3) Colecciones");
    //HashSet: sin orden, sin duplicados
    Set<String> hs = new HashSet<>();
    hs.add("A"); hs.add("B"); hs.add("C"); hs.add("A"); // duplicado ignorado
    System.out.println("HashSet: " + hs);

    //TreeSet: ordenado, sin duplicados (mostrar efecto de duplicados)
    Set<String> ts = new TreeSet<>();
    ts.add("G"); ts.add("E"); ts.add("E"); ts.add("A");
    System.out.println("TreeSet (ordenado, sin duplicados): " + ts);
}
```

4. ArrayList:
    a. arrayList con nombre de tus 5 ex.
    b. Eliminar 1 y agregar 1.

```java
// 4. ArrayList
static void arrayList() {
    titulo("4) ArrayList");
    ArrayList<String> ex = new ArrayList<>(Arrays.asList(
            "Montse", "Giselle", "Itzel", "Fernanda", "Andrea"));
    System.out.println("Inicial: " + ex);

    //Eliminar 1 y agregar 1
    ex.remove("Fernanda");
    ex.add("Maria fernanda");
    System.out.println("Actualizado: " + ex);
}
```

5. HashTable:
    a. Aplicar out.
    b. Contains.

```java
// 5. HashTable
static void hashTable() {
    titulo("5) Hashtable");
    Hashtable<Integer, String> tabla = new Hashtable<>();
    tabla.put(10, "Diez");
    tabla.put(20, "Veinte");
    tabla.put(30, "Treinta");

    //Aplicar out → imprimir el contenido
    System.out.println("Contenido: " + tabla);

    //Contains → containsKey / containsValue
    System.out.println("containsKey(20): " + tabla.containsKey(20));
    System.out.println("containsValue(\"Treinta\"): " + tabla.containsValue("Treinta"));
}
```

6. Mate:
    a. Valor absoluto.
    b. Máximo.
    c. Mínimo.
    d. Número aleatorio.
    e. Redondear.

```java
// 6. Mate (Math)
static void mate() {
    titulo("6) Mate (Math)");
    int x = -42, y = 17;
    System.out.println("abs(" + x + ") = " + Math.abs(x));
    System.out.println("max(" + x + ", " + y + ") = " + Math.max(x, y));
    System.out.println("min(" + x + ", " + y + ") = " + Math.min(x, y));

    double r = Math.random();
    System.out.println("random() = " + r);

    double v = 3.6;
    System.out.println("round(3.6) = " + Math.round(v));
}
```

```
=== 1) Arreglos ===
Inicial: [Santiago, Josue, Moises, Mafer]
A�adiendo Ari: [Santiago, Josue, Moises, Mafer, Ari]
Quitando Santiago: [Josue, Moises, Mafer, Ari]

=== 2) Cadenas ===
compareTo("hola", "holA") = 32
lastIndexOf('o') = 30
endsWith("tos") = true
substring(0, 11) = "programacio"

=== 3) Colecciones ===
HashSet: [A, B, C]
TreeSet (ordenado, sin duplicados): [A, E, G]

=== 4) ArrayList ===
Inicial: [Montse, Giselle, Itzel, Fernanda, Andrea]
Actualizado: [Montse, Giselle, Itzel, Andrea, Maria fernanda]

=== 5) Hashtable ===
Contenido: {10=Diez, 20=Veinte, 30=Treinta}
containsKey(20): true
containsValue("Treinta"): true

=== 6) Mate (Math) ===
abs(-42) = 42
max(-42, 17) = 17
min(-42, 17) = -42
random() = 0.3522868192299118
round(3.6) = 4
```

Conclusión

La práctica realizada constó de recopilar una serie de elementos del lenguaje Java, que reforzaron el uso de conocidos tipos de datos de Java, como estructuras de control, bibliotecas... De manera didáctica se analizó el menú desarrollado con vistas a aplicar lo aprendido: el uso de arreglos, cadenas, colecciones y algunas clases como ArrayList y Hashtable, trabajando así con datos de una manera más estructurada y flexible; la exploración de funciones matemáticas y el trabajo con fechas utilizando las clases Math, Date y Calendar, reafirmando la importancia de las bibliotecas como herramientas que facilitan la realización de tareas comunes.

Se implementaron todos los apartados requeridos, pero sí surgieron dudas en algunas secciones, con relación al uso de colecciones más avanzadas con algún método de las clases de cadenas, lo que pone de manifiesto que es necesario repetir con mayor detenimiento su funcionamiento. En términos generales, la práctica realizada ha supuesto un progreso en el conocimiento del control de flujo y la utilización de la biblioteca estándar de Java, un primer paso para encarar programas más complejos utilizando el paradigma de la programación orientada a objetos.

Referencias

- Facultad de Ingeniería UNAM. (2025). *Manual de prácticas de Programación Orientada a Objetos (MADO)*. Material de curso no publicado.

Recurso adicional

https://github.com/SARD82