	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): René Adrián Dávila Pérez

Asignatura: Programación Orientada a Objetos

Grupo: 1

No de Práctica(s): 1, 2

Integrante(s): 322184022

322238682

321570789

322069093

322249723

No. de lista o brigada: 5

Semestre: 2026-1

Fecha de entrega: 29/08/2025

Observaciones:

CALIFICACIÓN: _____

Índice

1 Introducción

Esta práctica aborda la implementación de tres algoritmos matemáticos: el cálculo factorial, la secuencia de Fibonacci y la conjetura de Collatz. El desafío principal consistió en desarrollar un programa que no solo resolviera cada problema correctamente, sino también proporcionara un menú de usuario interactivo, que permitiera la selección y ejecución de cada función. Para esta práctica es muy importante comprender y aplicar la recursividad, ya que permite resolver problemas complejos mediante la descomposición en instancias más simples del mismo problema.

Objetivos

- **Implementar funciones recursivas** para desarrollar la capacidad de resolver problemas complejos mediante dividiéndolos en sub-problemas más simples como lo permite la recursión.
- **Aplicar los conceptos matemáticos** (factorial, Fibonacci, Collatz) en un programa efectivo, ayudará a mejorar nuestras habilidades para resolver problemas (en estos casos, matemáticos) computacionalmente.

2 Marco Teórico

Previo a la práctica, fue necesario conocer y entender conceptos como algoritmo: es un conjunto de pasos (finitos y no ambiguos) ordenados que nos permiten resolver un problema. Utilizamos estructuras de control, que permiten decidir o repetir ciertas instrucciones, por ejemplo, los condicionales (if-else) y los ciclos (do-while). Por otra parte, implementamos funciones/métodos: bloques de código diseñados para cumplir instrucciones específicas.

- Método recursivo
  Un algoritmo recursivo es un método de programación en el que una función se llama a sí misma para resolver una versión más pequeña del mismo problema. Este proceso continúa hasta que el problema se vuelve lo suficientemente simple como para resolverlo directamente, sin más llamadas.
  La idea clave en la recursividad es que el problema se divide en partes más pequeñas y manejables, lo que facilita su resolución.[1]

- Factorial
  "El factorial de un número entero no negativo n, denotado como n!, se define como "el producto de todos los enteros positivos desde 1 hasta n. Por convención, el factorial de 0 es 1 " [4]
  Matemáticamente, se expresa como:

$$n! = n \times (n-1) \times (n-1) \times ... \times 1$$
$$\text{ó } n! = n(n-1)!$$

- Serie de Fibonacci
  "La serie se construye de la siguiente manera: dados con los números 0 y 1, cada número de la serie es sencillamente la suma de sus dos inmediato predecesores"[2]

- Collatz
  Secuencia definida de la siguiente forma: si el número es par se divide entre 2 y si el número es impar se multiplica por 3 y se le suma 1, se repite el proceso hasta llegar al número 1. [3]

3 Desarrollo

3.1 Menú

1. Primero se importó el "Scanner" para poder solicitar datos de entrada al usuario.

2. Después se desarrolló un menú con un ciclo "do-while" el cual se ejecuta siempre y cuando el usuario no introduzca el número 4.

3. Dentro del ciclo "do-while" se encuentra un "switch case" el cual cuenta con 4 opciones y un caso "default".

4. El caso uno contiene el Factorial, el caso dos la secuencia de Fibonacci, el caso tres Collatz, el cuatro la opción de salir y el "default" por si el usuario introduce una opción inexistente.

3.2 Factorial

Para el Factorial:

1. Primero se define la función y el número entero que va a recibir.

2. Se trabaja con `long` ya que los números factoriales pueden ser muy grandes.

3. Se crea una condición para los casos de 0 y 1, si se cumple se devuelve 1, en caso contrario se devuelve `n * factorial(n-1)`.

4. En el switch se le pide al usuario que introduzca un número entero y se vuelve a crear otra condición:

   - Si el usuario introduce un número negativo, menor a cero, el sistema le regresa un mensaje advirtiéndole que no es posible.
   - En caso contrario se llama a la función y se imprime el resultado.

5. Luego finaliza el case con "break".

3.3 Fibonacci

1. Se define la función la cual también regresará un número `long` y se le pasa el dato de entrada.
2. Se definen dos condiciones:
   - Si el número dado es 0 devuelve 0.
   - Si es 1 devuelve 1.
3. En caso contrario, se calcula recursivamente utilizando la fórmula de Fibonacci.
4. En el switch primero el usuario ingresa el número hasta el cual se desea calcular la secuencia.
5. Después se evalúa si el número es menor a cero; si es así, se le advierte al usuario que la secuencia no está definida para números negativos.
6. Si el número es válido, se utiliza un ciclo "for" para imprimir la secuencia desde el 0 hasta el número solicitado.
7. Luego finaliza el case con "break".

3.4 Collatz

1. Primero se define la función, en este caso se utiliza "void" ya que no será una función recursiva.
2. Después se imprime el número actual y se definen tres condiciones:
   - La primera es el caso base: si el número dado por el usuario es igual a 1, se termina la función usando `return`.
   - La segunda comprueba si es par; si el número es divisible entre dos, se divide entre dos.
   - La tercera: en caso de ser impar, el número será multiplicado por tres y sumado uno.
3. En el switch se le solicita al usuario que introduzca un número, se llama la función y se imprime en pantalla el resultado.
4. Luego finaliza el case con "break".

4 Resultados

```java
import java.util.Scanner;

public class Menu {

    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int opcion;

        do {
            // Mostrar menú
            System.out.println("\nMENÚ:");
            System.out.println("1. Factorial");
            System.out.println("2. Fibonacci");
            System.out.println("3. Secuencia de Collatz");
            System.out.println("4. Salir");
            System.out.print("Seleccione una opción (1-4): ");

            opcion = scanner.nextInt();

            switch (opcion) {
                case 1:


                case 2:


                case 3:


                case 4:


                default:
                    System.out.println("Opción no válida. Por favor, seleccione una opción del 1 al 4.");
            }

        } while (opcion != 4);

        scanner.close();
    }
```

Figura 1: Importación del "Scanner", estructura del código y menú

```
MENÚ:
1. Factorial
2. Fibonacci
3. Secuencia de Collatz
4. Salir
Seleccione una opción (1-4): 1
```

1.1. Menú en la terminal

```java
public static long factorial(int n) {
    if (n == 0 || n == 1) { // Casos base
        return 1;
    }
    return n * factorial(n - 1); // aplica recursividad con n-1
}
```

Figura 2: Función del factorial

```java
case 1:
    System.out.println("Ingrese el número para calcular su factorial: ");
    int numFactorial = scanner.nextInt();
    if (numFactorial < 0) {
        System.out.println("Error: No puede calcularse para numeros negativos");
    } else {
        long resultadoFactorial = factorial(numFactorial);
        System.out.println("El factorial de " + numFactorial + " es: " + resultadoFactorial);
    }
    break;
```

2.1. Factorial dentro del switch

```
MENÚ:
1. Factorial
2. Fibonacci
3. Secuencia de Collatz
4. Salir
Seleccione una opción (1-4): 1
Ingrese el número para calcular su factorial:
5
El factorial de 5 es: 120
```

2.2. Resultado del factorial

```java
public static long fibonacci(int n) {
    // Casos base 0 y 1
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    }
    //caso promedio
    return fibonacci(n - 1) + fibonacci(n - 2); // aplicando recursividad con las reglas de fibonacci
    }
```

Figura 3: Función Fibonacci

```java
case 2:
    System.out.print("Ingrese la posición en la secuencia de Fibonacci: ");
    int numFibonacci = scanner.nextInt();

    if (numFibonacci < 0) { // Control de excepciones:
        System.out.println("Error: La posición debe ser un número positivo");
    } else {
        //Implementación de for para mostrar secuencia completa
        for(int i=0;i<numFibonacci;i++){
            System.out.print(fibonacci(i)+ " ");
        }
    }
    break;
```

3.1. Fibonacci dentro del switch

```
MENÚ:
1. Factorial
2. Fibonacci
3. Secuencia de Collatz
4. Salir
Seleccione una opción (1-4): 2
Ingrese la posición en la secuencia de Fibonacci: 8
0 1 1 2 3 5 8 13
```

3.2. Resultado de Fibonaccil

```java
public static void collatz(int n){
    System.out.print(n + " ");
if (n == 1) { // Caso base
        return;
    } else if (n%2==0) { // si es par, se divide entre 2
        collatz(n/2);
    } else { // si es impar, se mult. por 3 y se le suma 1
        collatz(3*n+1);
    }

    }
}
```

Figura 4: Función Collatz

```java
case 3:
    System.out.print("Ingrese un numero para secuencia de Collatz: ");
    int numCollatz = scanner.nextInt();
    collatz(numCollatz);
    System.out.println();
    break;
```

4.1. Collatz dentro del switch

```
MENÚ:
1. Factorial
2. Fibonacci
3. Secuencia de Collatz
4. Salir
Seleccione una opción (1-4): 3
Ingrese un numero para secuencia de Collatz: 7
7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

4.2. Resultado de Collatz

5 Conclusión

Logramos comprobar que un problema matemático puede resolverse con un enfoque recursivo o de forma iterativa. La recursión nos permite realizar un código más simple y cercano a la definición matemática de cada función, aunque sus desventajas implican un mayor consumo de memoria y menor eficiencia en comparación con la forma iterativa, la elección por el "mejor"método depende del problema y de los recursos con los que contamos.

6 Referencias

[1] WsCube Tech. Recursive algorithm: Examples, complexity, types, uses. `https://www.wscubetech.com/resources/dsa/recursive-algorithm`, 2025. Consultado: 25-08-2025.

[2] Revista Digital Universitaria. Fibonacci y el número áureo. `https://www.revista.unam.mx/vol.6/num7/art68/art68-1.htm`, 2005. Consultado: 25-08-2025.

[3] Eric W. Weisstein. Collatz problem. https://mathworld.wolfram.com/CollatzProblem.html, 2023. Consultado: 25-08-2025.

[4] Eric W. Weisstein. Factorial. https://mathworld.wolfram.com/Factorial.html, 2023. Consultado: 25-08-2025.