	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: René Adrián Dávila Pérez

Asignatura: Programación Orientada a Objetos

Grupo: 01

No. de práctica(s): 01 y 02

Integrante(s):
322276824
322258516
425037384
320108116
322221415

No. de brigada: 01

Semestre: 2026-1

Fecha de entrega: 29 de agosto de 2025

Observaciones:

CALIFICACIÓN: _____

Índice

1. Introducción

- **Planteamiento del problema:**
  La familiaridad con el lenguaje de programación Java es crucial para entender el paradigma orientado a objetos. La implementación de tres algoritmos típicos, en dicho lenguaje: la conjetura de Collatz, la sucesión de Fibonacci (enfoque iterativo) y el cálculo de factorial (utilizando recursividad) junto con un menú interactivo representan problemas bastante completos para practicar las estructuras básicas de control y la sintaxis en este nuevo lenguaje de programación.

- **Motivación:**
  Aprender a implementar problemas en Java es de gran relevancia, ya que este lenguaje es ampliamente utilizado en la industria del software. Además, el entendimiento de conceptos básicos como las clases, los métodos y la diferencia entre la compilación y la ejecución del código en la Java Virtual Machine (JVM) es útil para sentar las bases de la programación orientada a objetos.

- **Objetivos:**
  Llevar a cabo una aproximación al lenguaje Java mediante la implementación de programas que resuelvan problemas matemáticos clásicos. Asimismo, comprender aspectos básicos del proceso de ejecución de un programa en Java y la aplicación de conceptos vistos en clases teóricas.

2. Marco Teórico

La Conjetura de Collatz, también conocida como el problema de $3x + 1$, se basa en una secuencia numérica definida por una sencilla regla matemática. De acuerdo con [2] la conjetura enuncia lo siguiente: «Para cualquier número entero positivo que se elija, si es par se divide entre 2, y si es impar se multiplica por 3 y se suma 1. Repitiendo este proceso sucesivamente, se llegará eventualmente al número 1.» Esto es:

$$f(n) = \begin{cases} \dfrac{n}{2}, & \text{si } n \text{ es par,} \\ 3n + 1, & \text{si } n \text{ es impar.} \end{cases}$$

Utilizando simulaciones computacionales a gran escala, se ha explorado el comportamiento de la Conjetura de Collatz. El análisis de estos datos ha revelado patrones y anomalías en las secuencias, lo que podría inspirar nuevas hipótesis para resolver este antiguo enigma matemático.

Por otro lado, la sucesión de Fibonacci es una secuencia numérica definida por la relación de recurrencia:

$$f(n) = \begin{cases} 0, & \text{si } n = 0, \\ 1, & \text{si } n = 1, \\ f(n-1) + f(n-2), & \text{si } n > 1. \end{cases}$$

Esta sucesión aparece en múltiples contextos matemáticos y naturales. Podemos observarla en la disposición de las hojas en los tallos de las plantas, en la formación de las ramas de los árboles, en la espiral de los girasoles y en el número de pétalos de ciertas flores. Además, esta proporción se encuentra en las espirales que forman los caparazones de moluscos como el Nautilus, lo que demuestra la presencia de este fenómeno matemático en la naturaleza. [1] En programación, se implementa comúnmente de forma recursiva o iterativa; en este caso, se utilizó la segunda.

Figura 1: Corte transversal de la concha de un Nautilus. [3]

Por último, el factorial de un número natural n se define como:

$$n! = n \cdot (n-1) \cdot (n-2) \cdots 1, \quad 0! = 1.$$

Un cálculo que se presta de manera natural a una implementación recursiva, ya que podemos enunciarlo como:

$$n! = \begin{cases} 1, & \text{si } n = 0, \\ n \cdot (n-1)!, & \text{si } n > 0. \end{cases}$$

El factorial de un número es un concepto matemático fundamental que tiene diversas aplicaciones en diferentes áreas, desde la combinatoria hasta la probabilidad, estadística y la programación. [4]

3. Desarrollo

Con el propósito de exponer de forma clara la lógica implementada en cada uno de los problemas, dividiremos la explicación en las siguientes secciones.

3.1. Conjetura de Collatz

Fuera del método principal `main` se creó el método `coll`, con modificador de acceso `public`, pudiendo accederse desde cualquier parte del programa, y con el modificador `static`, lo que permite invocarlo sin necesidad de crear un objeto. Al no devolver ningún valor, se declaró de tipo `void`, recibiendo como único parámetro un entero `n`, que representa el número sobre el cual se aplicará la conjetura de Collatz.
El funcionamiento del método se basa en el uso de condicionales `if`, aplicando el concepto de recursividad y considerando tres escenarios (los planteados en la sesión teórica):

1. Caso base: si el valor de `n` es 1, se imprime el número y finaliza la ejecución del método.

2. Caso par: se revisa si `n` es divisible entre dos con ayuda del operador módulo (`%`), se actualiza dividiéndolo entre dos y se realiza una nueva llamada recursiva a `coll`.

3. Caso impar: se actualiza mediante la operación $3n + 1$, y se vuelve a invocar el método hasta alcanzar el caso base.

3.2. Sucesión de Fibonacci

Para la implementación de la sucesión de Fibonacci se optó por un **enfoque iterativo**, priorizando la eficiencia computacional y el bajo consumo de memoria.
Fuera del método principal `main` se creó el método `fibo`. Al igual que `coll`, definido como `public static`. Este no devuelve ningún valor (`void`), ya que su función principal es imprimir la secuencia directamente en la consola. El método recibe un único número entero como parámetro (`n1`), que representa el número de términos de la sucesión que el usuario desea generar y visualizar.
La lógica del método parte de la inicialización de dos variables locales, `a` y `b`, con valores 0 y 1 respectivamente, correspondientes a los dos primeros términos de la sucesión. Posteriormente, se emplea un ciclo `for` que itera exactamente `n1` veces, y en cada paso:

1. Se imprime el valor actual de `a`.

2. Se calcula el siguiente término de la sucesión como `a + b`, almacenándolo en una variable auxiliar `siguiente`.

3. Se actualizan los valores: `a` toma el valor de `b` (avanzando un término en la secuencia) y `b` adopta el valor de `siguiente`.

El proceso se repite hasta completar los `n1` términos requeridos, tras lo cual el control regresa al método principal.

3.3. Factorial de un número

El cálculo del factorial se implementó mediante el método `factorial`, con modificador de acceso `public` y de tipo `static`, lo que permite invocarlo sin instanciar un objeto. Se le declaró de tipo `int` para que devuelva un entero al ejecutarse, recibe como parámetro un entero `n`, enviado desde el método principal.
La lógica del método sigue una estructura condicional `if-else` que considera dos casos (los planteados en la sesión teórica):

1. Caso base: se verifica si el parámetro es 0 o 1, se ejecuta la instrucción dentro del `if` y se regresa el valor de 1.

2. Caso general: para cualquier otro valor, se retorna el producto de `n` por la llamada recursiva del mismo método enviándole como parámetro `n-1`, haciendo las llamadas recursivas necesarias hasta devolver el valor de `n` actualizado.

3.4. Menú de opciones

El control del programa se centralizó en el método `main`, dentro del cual se declaró un ciclo `while` que mantiene activo el menú hasta que el usuario seleccione la opción de salida (4). Se imprimen las cuatro opciones disponibles para el usuario: cálculo de factorial, sucesión de Fibonacci, conjetura de Collatz y la opción de terminar la ejecución.
La elección del usuario se captura mediante un objeto de la clase `Scanner` en una variable `opcion` y a partir de ahí se emplea una estructura `switch-case` que dirige el flujo hacia el método correspondiente.

4. Resultados

Menú que presenta las opciones de cálculo de factorial, sucesión de Fibonacci, conjetura de Collatz y salir.

Figura 2: Menú de opciones.

Seleccionando la opción para calcular el factorial de un número (1).

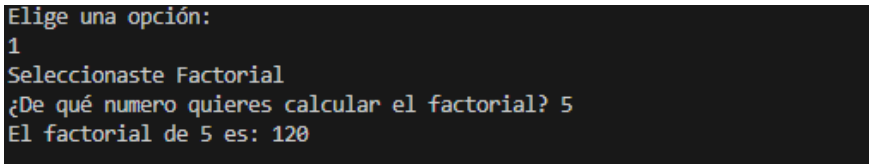

Figura 3: Factorial de un valor, en este caso 5.

Seleccionando la opción para mostrar la sucesión de Fibonacci (2).

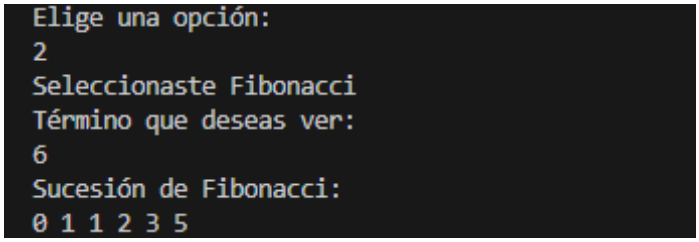

Figura 4: Primeros 6 términos de la sucesión de Fibonacci.

Seleccionando la opción para la Conjetura de Collatz (3).

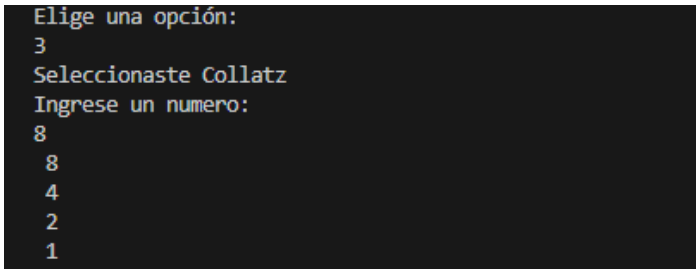

Figura 5: Desarrollo de la Conjetura de Collatz para un valor par.

Seleccionando nuevamente la opción para la Conjetura de Collatz (3).

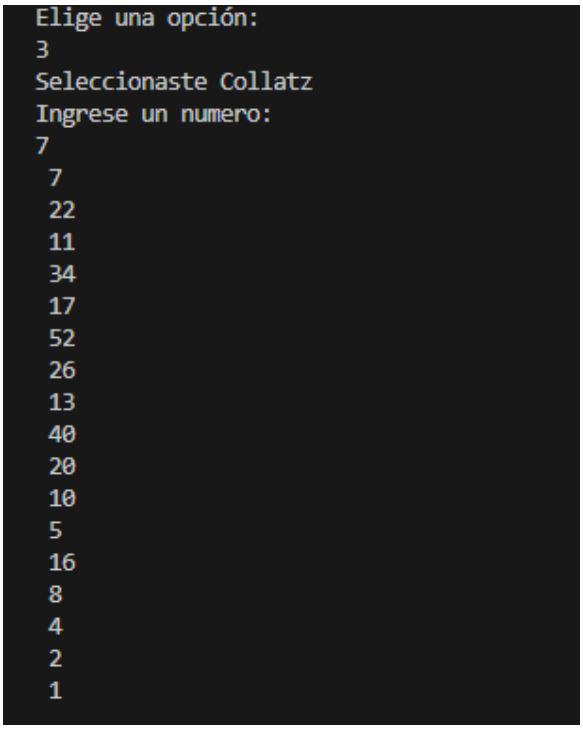

Figura 6: Desarrollo de la Conjetura de Collatz para un valor impar.

Seleccionando la opción para salir del programa (4).

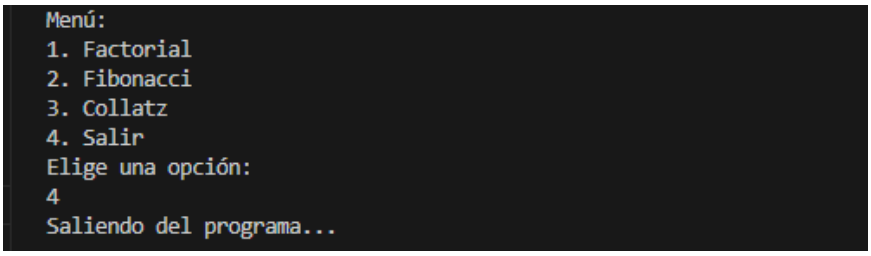

Figura 7: Finalizando programa.

5. Conclusiones

La implementación de los algoritmos de Collatz, Fibonacci y factorial en Java representó un ejercicio integral que permitió pasar de la comprensión teórica a la práctica en código ejecutable. Con ello se reforzó el dominio de la sintaxis básica del lenguaje y se afianzaron conceptos clave como el control de flujo, la recursividad, la iteración y el diseño modular.
La recursividad resultó adecuada para problemas con definiciones autoreferenciales, como el factorial y la conjetura de Collatz, mientras que el enfoque iterativo aplicado a Fibonacci mostró mayor eficiencia para secuencias lineales.
Además, la integración de los tres métodos en un programa con menú interactivo demostró cómo organizar de manera modular componentes independientes en un sistema cohesivo, resaltando la importancia de comprender el ciclo de compilación y ejecución propio de Java a través del bytecode y la JVM.

Referencias

[1] *Fibonacci*. Ago. de 2025. URL: https://arquimedes.matem.unam.mx/mati/actividades/actividad_fibonacci/index.html.

[2] *La Conjetura de Collatz*. Ago. de 2025. URL: https://www.teoremas.club/la-conjetura-de-collatz-por-que-incluso-despues-de-decadas-esta-simple-conjetura-sigue-siendo-un-misterio-sin-resolver/.

[3] Light Mushroom. «Nautilus». En: (2016).

[4] *Qué es un Factorial de un Número*. Ago. de 2025. URL: https://gesnomic.com/que-es-un-factorial-de-un-numero/.