	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): Dávila Pérez René Adrián

Asignatura: Programación Orientada a Objetos

Grupo: 1

No de Práctica(s): 4

Integrante(s): 322089020

322089817

322151194

425091586

322085390

No. de lista o brigada: Equipo 4

Semestre: 2026-1

Fecha de entrega: 19 de septiembre de 2025

Observaciones:

CALIFICACIÓN: _____

Índice

1. Introducción

Planteamiento del Problema

El ejercicio consiste en desarrollar un programa en Java que reciba como parámetros las coordenadas de dos puntos en el plano cartesiano y calcule la distancia que existe entre ellos. Para lograr esto, se hace uso de la fórmula de distancia euclidiana, implementada dentro de una clase controladora, y se muestra el resultado mediante una interfaz gráfica que utiliza una ventana con un botón de interacción.

Motivación

La práctica busca reforzar el conocimiento de los conceptos teóricos vistos en clase, específicamente en lo relacionado con la programación orientada a objetos, el manejo de clases y objetos, y la implementación de interfaces gráficas en Java. El uso de ventanas interactivas permite dar un enfoque más visual e intuitivo al problema, demostrando cómo los cálculos matemáticos pueden integrarse de manera amigable con el usuario.

Objetivos

El objetivo principal de esta práctica es aplicar los conceptos de la programación orientada a objetos en Java para resolver un problema geométrico y presentar la solución de forma interactiva.

- Implementar clases con atributos y métodos que representen puntos y cálculos geométricos.
- Utilizar el método main para recibir entradas externas y transformarlas en objetos de tipo Punto.
- Crear una interfaz gráfica sencilla con Swing para mostrar al usuario el resultado del cálculo.
- Reforzar el vínculo entre teoría matemática y su aplicación práctica en un programa funcional.

2. Marco Teórico

El Plano Cartesiano y la Distancia entre Dos Puntos

El plano cartesiano es un sistema de coordenadas bidimensional en el que cada punto se representa mediante un par ordenado (x, y). Para calcular la distancia entre dos puntos $P_0(x_0, y_0)$ y $P_1(x_1, y_1)$, se emplea la siguiente fórmula:

$$d = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

Dicha fórmula proviene del teorema de Pitágoras, ya que la diferencia entre las coordenadas de los puntos forma los catetos de un triángulo rectángulo, mientras que la distancia representa la hipotenusa. Este cálculo es ampliamente utilizado en matemáticas aplicadas, física e ingeniería.[5]

Representación de un Punto

Un punto en el plano cartesiano se modela mediante dos coordenadas, x y y, que indican su posición en los ejes horizontal y vertical respectivamente. En la programación, se acostumbra encapsular estas coordenadas dentro de una estructura u objeto para poder manipularlas de manera ordenada, lo que permite realizar cálculos como la distancia o transformaciones geométricas de manera eficiente.[1]

Bibliotecas Utilizadas

Para el desarrollo del programa se emplearon distintas bibliotecas que amplían las capacidades del lenguaje:

- **java.lang.Math**: permite realizar operaciones matemáticas avanzadas mediante funciones como `sqrt()` (raíz cuadrada) y `pow()` (potencia). [4]
- **javax.swing**: proporciona los componentes necesarios para construir una interfaz gráfica de usuario (ventanas, botones, cuadros de diálogo).[2]

- **java.awt.event**: se encarga de gestionar los eventos generados por el usuario, como el clic en un botón, para que el programa ejecute la acción correspondiente. [3]

Interfaz Gráfica e Interacción con el Usuario

El uso de interfaces gráficas en lugar de la consola mejora la experiencia del usuario, ya que facilita la interacción a través de ventanas y botones. En este caso, la interfaz permite al usuario obtener el cálculo de la distancia de manera clara y visual, mostrando los resultados en un cuadro emergente (`JOptionPane`).

3. Desarrollo

Implementación del Código

El programa simula una ventana que determina la distancia entre dos puntos aportados en la misma ejecución del código. Hacemos uso de las siguientes librerías: *java.lang.Math* (para poder operar la raíz cuadrada); *javax.swing* (permite habilitar el entorno gráfico al código, con él podemos hacer uso de botones, ventanas y campos de texto); *java.awt.event* (gestiona las acciones externas que produce el usuario, determina que acción se llevará a cabo si el usuario presiona una tecla o si hace clic en un botón, es una librería que puede complementar el entorno gráfico).

Archivos

Main

El método principal se reciben los argumentos de las coordenadas de los puntos separados por un espacio, estos se guardan como atributos en un objeto de la clase *Punto* para almacenarse de forma ordenada por ejemplo: $P_0(x_0, y_0), P_1(x_1, y_1)$. Posteriormente se crea un objeto de la clase *Mensajes* sin atributos para después mostrar correctamente los resultados. Además se crea un objeto de la clase *Ventana* que servirá para mostrar los resultado de forma externa en una ventana a este objeto se le dan atributos que son: el objeto para ver el resultado de la operación y las coordenadas x y y de cada uno de los puntos dados. Y por último se visualiza la ventana de los resultados.

Punto

Esta clase modela un punto en el plano cartesiano con dos atributos, x y y, que por defecto valen 1, tiene un constructor vacío que crea un punto $(1, 1)$ y otro constructor que permite asignar valores personalizados a las coordenadas. Además, sobrescribe el método *toString()* para que, al imprimir el objeto, se muestre en el formato "$Punto(x = ..., y = ...)$".

Ventana

Esta clase hereda de *JFrame* y crea una interfaz gráfica sencilla que muestra un botón. Al instanciarse, recibe un objeto *Mensajes* y las coordenadas de dos puntos (x_0, y_0 y x_1, y_1). En el constructor se configuran las propiedades de la ventana (título, tamaño, cierre y posición) y se añade un botón con el texto *"Haz clic aquí para ver el resultado"* que cuando el usuario pulsa ese botón, se ejecuta un *ActionListener* que usa el controlador *Mensajes* para calcular la distancia y generar un mensaje con los valores de los puntos, y luego lo muestra en una ventana emergente *(JOptionPane)*.

Mensajes

Esta clase con un solo método recibe las coordenadas x y y de cada punto para calcular la distancia entre ellos. El método retorna un *String* para visualizar los puntos y la distancia entre estos usando la fórmula $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$, haciendo uso de *sqrt() y pow()*, funciones de *Math*.

Pruebas

```
Input(Args): 3 3 3 -3
Output(En ventana):
La distancia entre P0 (3.0, 3.0) y P1 (3.0, -3.0) es 6.0 unidades de distancia
Input(Args): -5 2 6 -3
Output(En ventana):
La distancia entre P0 (-5.0, 2.0) y P1 (6.0, -3.0) es 12.08... unidades de distancia
```

4. Resultados

Debemos de ingresar los argumentos junto a la ejecución de nuestro documento para poder ejecutar el código. Se ponen dos pares de números enteros que representarán las coordenadas de nuestros puntos, el código se encargará de determinar su distancia a partir de las operaciones que habilitamos con la librería .math. Recordemos que al ser un código con la ausencia de Scanner los argumentos van directamente después de la ejecución del código .java.

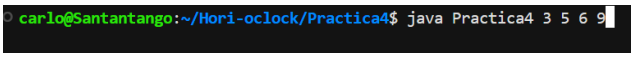

Figura 1: Se ejecuta el código junto con las coordenas de los puntos.

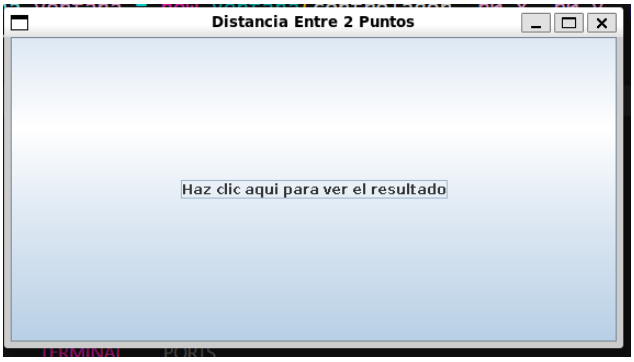

Figura 2: Posteriormente emerge una ventana con las siglas "Haz clic para ver el resultado"

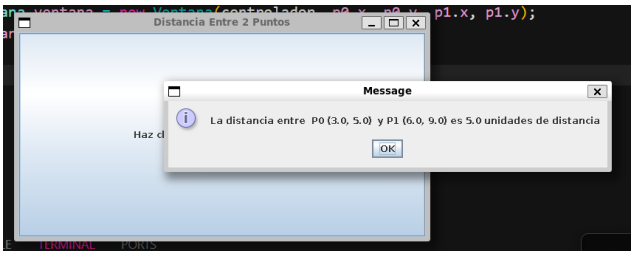

Figura 3: Al hacer clic en la frase nos saldrá un nuevo campo de texto con la distancia entre ambos puntos, siendo el resultado que buscabamos calcular.

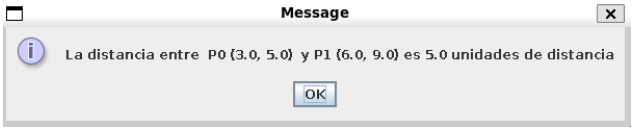

Figura 4: Se muestra mejor la nueva ventana con el resultado.

5. Conclusiones

El desarrollo de esta práctica permitió aplicar de manera efectiva los conceptos de programación orientada a objetos, junto con el manejo de interfaces gráficas en Java. Se cumplió con el objetivo de calcular y mostrar la distancia entre dos puntos, integrando conocimientos matemáticos con herramientas de programación.

Asimismo, se logró evidenciar cómo la teoría se traduce en práctica: desde la representación de puntos en un plano hasta el uso de la fórmula de distancia, todo encapsulado en un programa interactivo que facilita la experiencia del usuario.

Por último, esta práctica reafirma la importancia de combinar el razonamiento lógico-matemático con las herramientas de programación, demostrando que conceptos abstractos pueden transformarse en aplicaciones funcionales y amigables.

Referencias

[1] Baeldung. *Calculate the Distance Between Two Points in Java*. 2025. URL: https://www.baeldung.com/java-distance-between-two-points.

[2] GeeksforGeeks. *Introduction to Java Swing*. 2025. URL: https://www.geeksforgeeks.org/java/introduction-to-java-swing/.

[3] Oracle. *Java Platform SE 8: java.awt.event.ActionListener Documentation*. 2014. URL: https://docs.oracle.com/javase/8/docs/api/java/awt/event/ActionListener.html.

[4] Oracle. *Java Platform SE 8: java.lang.Math Documentation*. 2014. URL: https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html.

[5] Oracle. *Java SE 17 & JDK 17: java.lang.Math.sqrt(double) Method Documentation*. 2023. URL: https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html#sqrt(double).