

# Informe de Ejercicios teoría 03

## Tema: Clases

**Nota**

Estudiante	Escuela	Asignatura
Mikhail Gabino Velasque Arcos mvelasquea@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Teoría FUNDAMENTOS DE LA PROGRAMACION II Semestre: II Código: 20214260

Laboratorio	Tema	Duración
03	Resolución de los problemas de teoría FP2	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 22 de diciembre 2023	Al 23 de diciembre 2023

## 1. Tarea

- Ejercicio 01: Crear una clase base denominada Punto que conste de las coordenadas x e y. A partir de esta clase, definir una clase denominada Circulo que tenga las coordenadas del centro y un atributo denominado radio. Entre las funciones miembro de la primera clase, deberá existir una función distancia() que devuelva la distancia entre dos puntos, donde:  $\text{Distancia} = ((x_2 - x_1)^2 + (y_2 - y_1)^2)^{1/2}$
- Ejercicio 02: Utilizando la clase construida en el ejercicio 01, obtener una clase derivada Cilindro derivada de Circulo. La clase Cilindro deberá tener una función miembro que calcule la superficie de dicho cilindro. La fórmula que calcula la superficie del cilindro es  $S = 2r(l + r)$  donde r es el radio del cilindro y l es la longitud.
- Caso de estudio especial: herencia múltiple. Es un tipo de herencia en la que una clase hereda el estado (estructura) y el comportamiento de más de una clase base. (hay herencia múltiple cuando una clase hereda de más de una clase). Java no permite la herencia múltiple, pero se puede conseguir la implementación de la herencia múltiple usando interfaces. Implemente el siguiente diagrama de clases UML y consiga pruebas válidas.

## 2. Equipos, materiales y temas utilizados

- Git , Git hub , clases, Diagramas UML ,herencia , herencia multiple
- VIM 9.0.

- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.

### 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- URL :<https://github.com/mvelasquea/fp2-23b.git>

## 4. Ejercicio 1: CREACION DE LAS CLASES PUNTO Y MAIN

### 4.1. Creando la clase Punto/operaciones y la clase PRINCIPAL

Listing 1: Creando la clase Punto

```
package problemas_ejercicio22_diciembre;
import java.util.*;
/*
    Ejercicio de teoria ( ejercicio 1)
    > clase punto
    Autor :Mikhail Gabino Velasque Arcos
    colaboro:---
    tiempo:horas
    */
//creacion de la clase punto para acoplarlo a la clase principal llamada problema1_mains
public class Punto {
    int x, y;
    public Punto(int x, int y) {
        this.x = x;
        this.y = y;
    }
    //funcion que aplica la forma de distancia entre puntos haciendo el uso de math ,
    pow(potencia), sqrt(raiz)
    // tomando los valores ya ingresados por el usuario
    public double distancia(Punto otroPunto) {
        return Math.sqrt(Math.pow((otroPunto.x - this.x), 2) + Math.pow((otroPunto.y -
            this.y), 2));
    }
}
```

Listing 2: Creando la clase Main

```
package problemas_ejercicio22_diciembre;
import java.util.*;
/*
    Ejercicio de teoria ( ejercicio 1)
    > clase PRINCIPAL MAIN
```

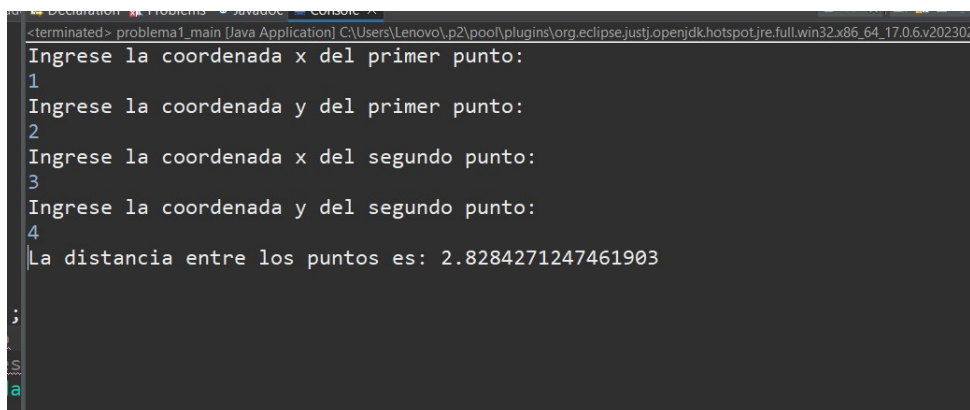
```
Autor :Mikhail Gabino Velasque Arcos
colaboro:---
tiempo:horas
*/
public class problema1_main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
// introduccion de valores tanto del punto A como el del punto B
        //Se obtiene 2 valores "x" y dos valores "y" respectivamente a los puntos ya antes
        mencionados
        System.out.println("Ingrese la coordenada x del primer punto: ");
        int x1 = scanner.nextInt();
        System.out.println("Ingrese la coordenada y del primer punto: ");
        int y1 = scanner.nextInt();

        Punto punto1 = new Punto(x1, y1);

        System.out.println("Ingrese la coordenada x del segundo punto: ");
        int x2 = scanner.nextInt();
        System.out.println("Ingrese la coordenada y del segundo punto: ");
        int y2 = scanner.nextInt();

        Punto punto2 = new Punto(x2, y2);
// se llama a la funcion de "distancia" para luego ejecutarla y mostrar el resultado
        System.out.println("La distancia entre los puntos es: " + punto1.distancia(punto2));
    }
}
```

## 4.2. Resultados



```
<terminated> problema1_main [Java Application] C:\Users\Lenovo\AppData\Local\Temp\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v202302
Ingrese la coordenada x del primer punto:
1
Ingrese la coordenada y del primer punto:
2
Ingrese la coordenada x del segundo punto:
3
Ingrese la coordenada y del segundo punto:
4
La distancia entre los puntos es: 2.8284271247461903
```

- Se muestra como el programa pide valores al usuario para completar el valor del punto 1 y del punto 2 para aplicar la función dentro de la clase punto (Distancia entre dos puntos)

## 5. Ejercicio 2: CREACION DE LAS CLASES Cilindro Y Main

### 5.1. Creando la clase Punto/operaciones y la clase PRINCIPAL

Listing 3: Creando la clase cilindro

```
package problemas_ejercicio22_diciembre;
```

```
/*Ejercicio de teoria ( ejercicio 2)
> class Cilindro
Autor :Mikhail Gabino Velasque Arcos
colaboro:---
tiempo:horas
*/
// Clase derivada Cilindro de la clase Punto
public class Cilindro extends Punto {
    private double longitud; // Longitud del cilindro

    // Constructor que toma dos puntos y establece la longitud del cilindro
    public Cilindro(Punto punto1, Punto punto2, double longitud) {
        super(punto1.x, punto1.y); // Llama al constructor de la clase base (Punto) para
        establecer las coordenadas del punto
        this.longitud = longitud;
    }
    // Funcin para calcular la superficie del cilindro
    public double calcularSuperficie() {
        double radio = distancia(new Punto(0, 0)); // Usa la distancia entre el punto y el
        origen como el radio
        return 2 * Math.PI * radio * (longitud + radio);
    }
}
}}
```

Listing 4: Creando la clase Main

```
import java.util.Scanner;
/*Ejercicio de teoria ( ejercicio 2)
> class main
Autor :Mikhail Gabino Velasque Arcos
colaboro:---
tiempo:horas
*/
public class problema2_main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        //solicitando valores de los puntos al usuario al igual que la longitud
        System.out.println("Ingrese la coordenada x del primer punto: ");
        int x1 = scanner.nextInt();
        System.out.println("Ingrese la coordenada y del primer punto: ");
        int y1 = scanner.nextInt();

        Punto punto1 = new Punto(x1, y1);

        System.out.println("Ingrese la coordenada x del segundo punto: ");
        int x2 = scanner.nextInt();
        System.out.println("Ingrese la coordenada y del segundo punto: ");
        int y2 = scanner.nextInt();

        Punto punto2 = new Punto(x2, y2);

        System.out.println("Ingrese la longitud del cilindro: ");
        double longitudCilindro = scanner.nextDouble();
    }
}
```

```
// creando el objeto de cilindro
Cilindro cilindro = new Cilindro(punto1, punto2, longitudCilindro);

// manda valores a la clase cilindro y lo calcula para mostrarlo
System.out.println("La superficie del cilindro es: " + cilindro.calcularSuperficie());
}
}
```

## 5.2. Resultados

```
<terminated> problema2_main [Java Application] C:\Users\Lenovo\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full
Ingrese la coordenada x del primer punto:
1
Ingrese la coordenada y del primer punto:
2
Ingrese la coordenada x del segundo punto:
3
Ingrese la coordenada y del segundo punto:
4
Ingrese la longitud del cilindro:
4
La superficie del cilindro es: 87.61444438422375
```

Se muestra que con ayuda del código del anterior ejercicio usa el valor ya encontrado (radio) o la distancia entre 1 punto al otro, aplica la fórmula ya dada y ejecuta mostrando el resultado

```
teoria/
|--- Ejercicio1.java
|--- Ejercicio2.java
|--- Ejercicio3.java
|--- gitignore.java

|--- latex
|   |--- img
|       |--- logo_abet.png
|       |--- logo_episunsa.png
|       |--- logo_unsa.jpg
|       |--- captura1.png
|       |--- captura2.png

|--- latex_ejercicio3_teoría_fp2.0.pdf
|--- latex_ejercicio3_teoría_fp2.0.tex
|--- src
|   |--- ejercicio1.java
|   |--- ejercicio2.java
|   |--- ejercicio3.java
```

## 6. Ejercicio 3: CREACION DE una clase UML

### 6.1. Creando un diagrama UML donde se usa una herencia multiple con el grafico ya mostrado

