



Instituto Politécnico Nacional

Escuela Superior de Cómputo



Unidad de académica: Paradigmas de programación

Actividad: “POO sobrecarga de métodos”

Equipo:

- Elizalde Hernández Alan
- Reyes Ruíz Yoselyn Estefany
- Solares Velasco Arturo Misael
 - Solís Lugo Mayra
- Toral Hernández Leonardo Javier

Grupo: 3CV1

Profesor: García Floriano Andrés

Fecha:

13 de mayo de 2024

Parte I

Diseña y desarrolla una Clase llamada Punto3D, la cual tendrá tres atributos privados

Atributos x, y, z privados con valor default

```
class Punto3D {  
    private double x;  
    private double y;  
    private double z;  
  
    public punto3D() {  
        this.x = 0.0;  
        this.y = 0.0;  
        this.z = 0.0;  
    }  
  
    public Punto3D(double x, double y, double z) {  
        this.x = x;  
        this.y = y;  
        this.z = z;  
    }  
}
```

Métodos set

```
    public void setX(double x) {  
        this.x = x;  
    }  
  
    public void setY(double y) {  
        this.y = y;  
    }  
  
    public void setZ(double z) {  
        this.z = z;  
    }  
}
```

Metodos get

```
public double getX() {  
    return x;  
}  
  
public double getY() {  
    return y;  
}  
  
public double getZ() {  
    return z;  
}
```

Calculo de la distancia con la formula euclidiana

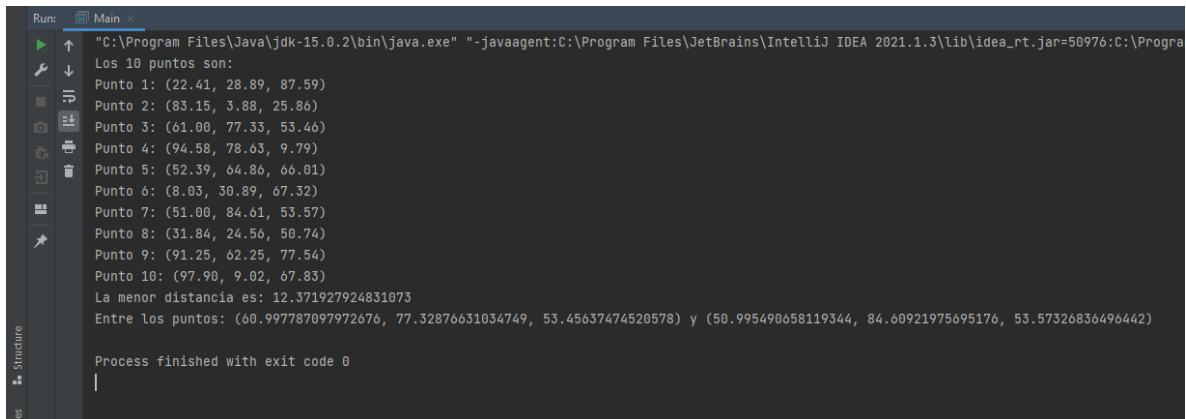
$$distancia = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$

```
public double distancia(Punto3D otroPunto) {  
    double dx = this.x - otroPunto.getX();  
    double dy = this.y - otroPunto.getY();  
    double dz = this.z - otroPunto.getZ();  
    return Math.sqrt(dx * dx + dy * dy + dz * dz);  
}  
}
```

Parte II

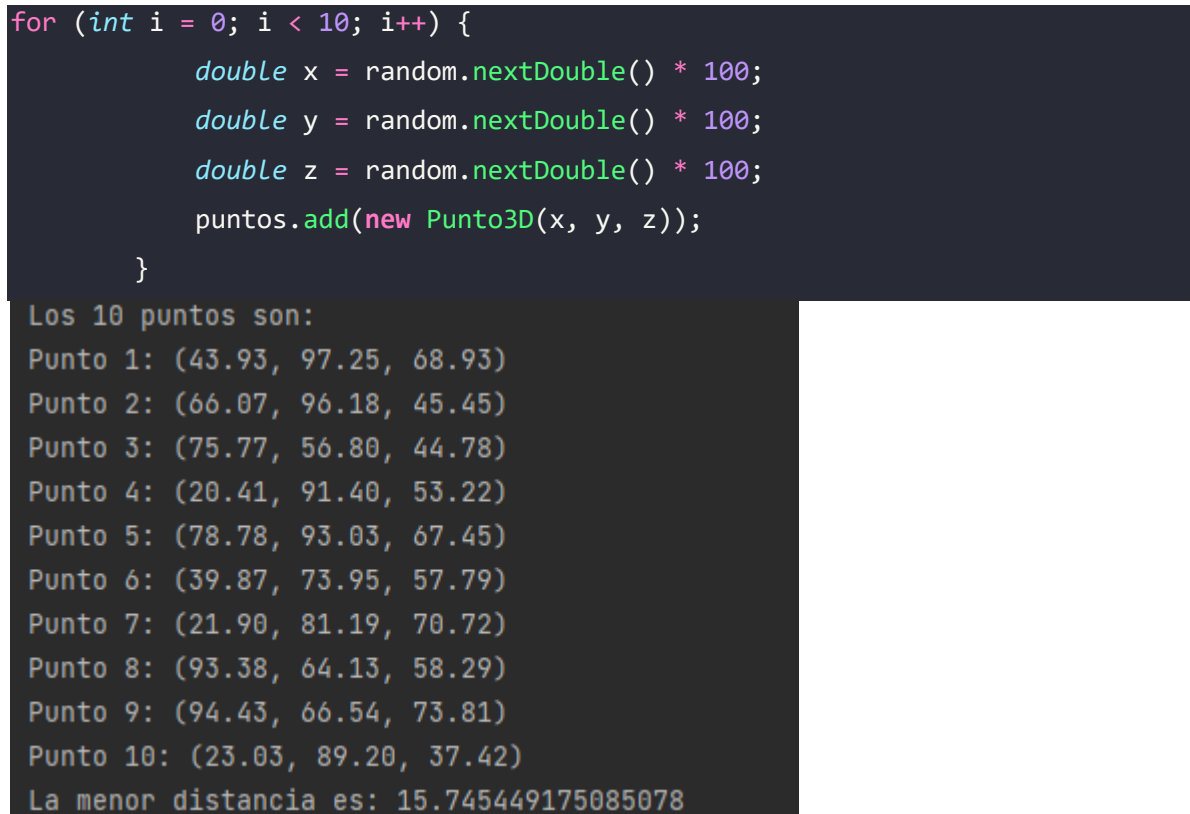
Ejecución del código

Crea un arreglo de 10 objetos de tipo Punto3D, calcula la distancia entre todos los objetos y determina la menor de las distancias obtenidas.



```
Run: Main
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.1.3\lib\idea_rt.jar=50976:C:\Progra
Los 10 puntos son:
Punto 1: (22.41, 28.89, 87.59)
Punto 2: (83.15, 3.88, 25.86)
Punto 3: (61.00, 77.33, 53.46)
Punto 4: (94.58, 78.63, 9.79)
Punto 5: (52.39, 64.86, 66.01)
Punto 6: (8.03, 30.89, 67.32)
Punto 7: (51.00, 84.61, 53.57)
Punto 8: (31.84, 24.56, 50.74)
Punto 9: (91.25, 62.25, 77.54)
Punto 10: (97.90, 9.02, 67.83)
La menor distancia es: 12.371927924831073
Entre los puntos: (60.997787097972676, 77.32876631034749, 53.45637474520578) y (50.995490658119344, 84.60921975695176, 53.57326836496442)
Process finished with exit code 0
```

Primero creamos los 10 objetos y para hacerlo diferente en cada ejecución se aleatorizan los valores de cada punto



```
for (int i = 0; i < 10; i++) {
    double x = random.nextDouble() * 100;
    double y = random.nextDouble() * 100;
    double z = random.nextDouble() * 100;
    puntos.add(new Punto3D(x, y, z));
}

Los 10 puntos son:
Punto 1: (43.93, 97.25, 68.93)
Punto 2: (66.07, 96.18, 45.45)
Punto 3: (75.77, 56.80, 44.78)
Punto 4: (20.41, 91.40, 53.22)
Punto 5: (78.78, 93.03, 67.45)
Punto 6: (39.87, 73.95, 57.79)
Punto 7: (21.90, 81.19, 70.72)
Punto 8: (93.38, 64.13, 58.29)
Punto 9: (94.43, 66.54, 73.81)
Punto 10: (23.03, 89.20, 37.42)
La menor distancia es: 15.745449175085078
```

Encontrar la menor distancia

Se usa el algoritmo de búsqueda exhaustiva dado que solo son 10 objetos, por lo que, la complejidad no es demasiada y permite que busquemos con fuerza bruta

```
double menorDistancia = Double.POSITIVE_INFINITY;
Punto3D punto1 = null;
Punto3D punto2 = null;

for (int i = 0; i < puntos.size(); i++) {
    for (int j = i + 1; j < puntos.size(); j++) {
        double distanciaActual =
puntos.get(i).distancia(puntos.get(j));
        if (distanciaActual < menorDistancia) {
            menorDistancia = distanciaActual;
            punto1 = puntos.get(i);
            punto2 = puntos.get(j);
        }
    }
}
```

La menor distancia es: 6.2894214067222185
Entre los puntos: (4.447764975456414, 13.138412558014656, 72.32963888075317)
y
(3.607527467520033, 8.962933460877487, 76.95740258160946)

Conclusión

La practica nos permite entender constructores sobrecargados, y su implementación en Java, así como el uso de arraylist para tener arreglos de objetos.

También el uso de conocimientos previos de las practicas como los principios de encapsulación y abstracción.

En general, el conocimiento adquirido para desarrollar la clase 'punto3D' nos permite entender los principios aplicados de la programación orientada a objetos en adición con los conocimientos de otras materias y su relación con problemas en el mundo real de manera efectiva.

Anexos

Codigo

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

class Punto3D {
    private double x;
    private double y;
    private double z;

    public Punto3D(double x, double y, double z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }

    // Métodos set
    public void setX(double x) {
        this.x = x;
    }

    public void setY(double y) {
        this.y = y;
    }

    public void setZ(double z) {
        this.z = z;
    }

    // Métodos get
```

```

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public double getZ() {
        return z;
    }

    // Distancia euclidiana entre dos puntos
    public double distancia(Punto3D otroPunto) {
        double dx = this.x - otroPunto.getX();
        double dy = this.y - otroPunto.getY();
        double dz = this.z - otroPunto.getZ();
        return Math.sqrt(dx * dx + dy * dy + dz * dz);
    }
}

public class Main {
    public static void main(String[] args) {
        List<Punto3D> puntos = new ArrayList<>();
        Random random = new Random();

        // Crear los 10 objetos de tipo Punto3D con valores
        aleatorios
        for (int i = 0; i < 10; i++) {
            double x = random.nextDouble() * 100;
            double y = random.nextDouble() * 100;

```

```

        double z = random.nextDouble() * 100;
        puntos.add(new Punto3D(x, y, z));
    }

    // Imprimir los puntos
    System.out.println("Los 10 puntos son:");
    for (int i = 0; i < puntos.size(); i++) {
        Punto3D punto = puntos.get(i);
        System.out.printf("Punto %d: (%.2f, %.2f, %.2f)%n", i
+ 1, punto.getX(), punto.getY(), punto.getZ());
    }

    // Calcular todas las distancias y encontrar la menor
    double menorDistancia = Double.POSITIVE_INFINITY;
    Punto3D punto1 = null;
    Punto3D punto2 = null;

    for (int i = 0; i < puntos.size(); i++) {
        for (int j = i + 1; j < puntos.size(); j++) {
            double distanciaActual =
puntos.get(i).distancia(puntos.get(j));
            if (distanciaActual < menorDistancia) {
                menorDistancia = distanciaActual;
                punto1 = puntos.get(i);
                punto2 = puntos.get(j);
            }
        }
    }

    System.out.println("La menor distancia es: " +
menorDistancia);

```



```
        System.out.println("Entre los puntos: (" + punto1.getX() +  
        ", " + punto1.getY() + ", " + punto1.getZ() + ") y (" +  
            punto2.getX() + ", " + punto2.getY() + ", " +  
punto2.getZ() + ")");  
    }  
}
```