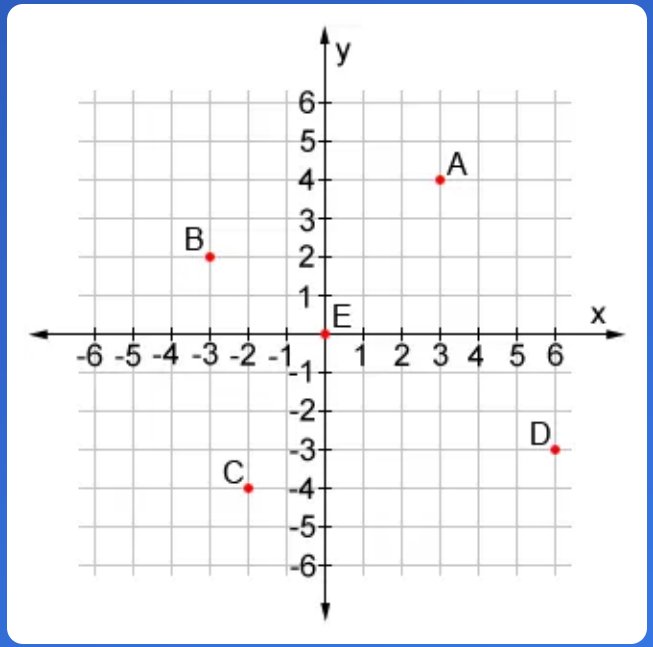
Resolução da Questão 2

Classe Ponto2D

Enunciado:

Crie uma classe chamada **Ponto2D** para representar um ponto no plano cartesiano com as seguintes características:

- Atributos privados **x** e **y** do tipo double.
- Construtor default que inicializa o ponto na origem (0,0).
- Construtor que recebe as coordenadas x e y como parâmetros.
- Métodos getters para os atributos x e y.
- Método **calcularDistancia** que recebe outro ponto como parâmetro e retorna a distância euclidiana entre os dois pontos.
- Método **toString** que retorna uma representação textual do ponto no formato "(x, y)".
- Crie uma classe de teste para verificar o funcionamento da classe Ponto2D.



Estrutura da Classe Ponto2D

Atributos e métodos

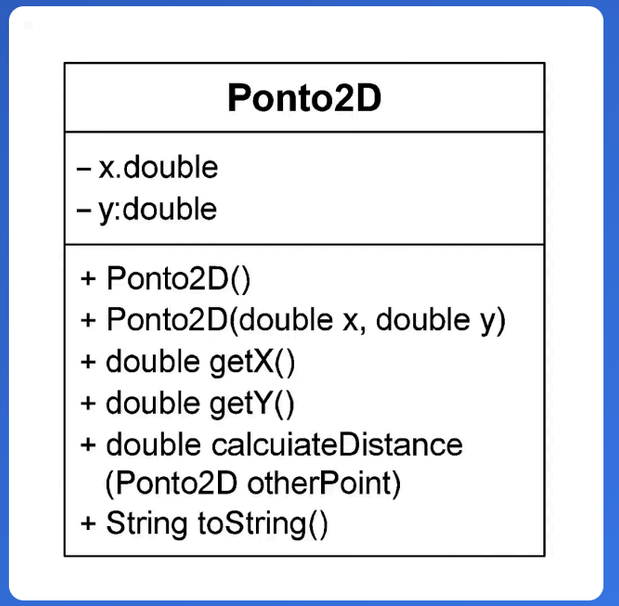
Implementação da Classe:

```
public class Ponto2D {
    private double x;
    private double y;

    // Construtor default
    public Ponto2D() {
        this(0.0, 0.0);
    }

    // Construtor com parâmetros
    public Ponto2D(double x, double y) {
        this.x = x;
        this.y = y;
    }

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



Características:

- Atributos privados (x, y) para coordenadas
- Construtores: default e parametrizado
- Métodos getters para acesso aos atributos

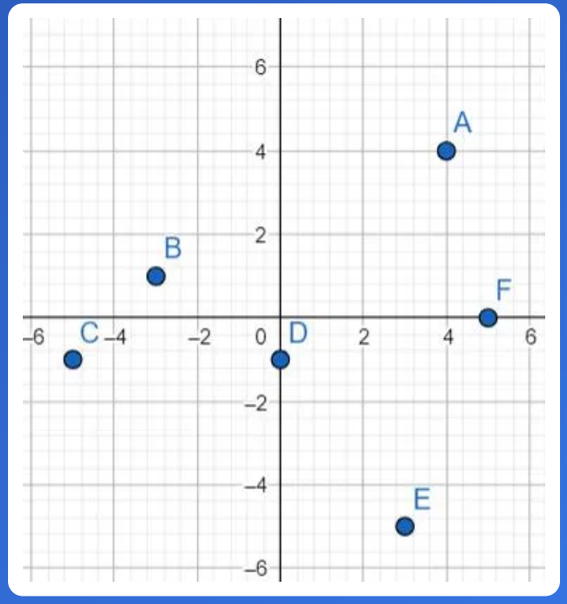
Construtores da Classe

Inicialização de objetos Ponto2D

Implementação dos Construtores:

```
// Construtor default (i)
public Ponto2D() {
    this(0.0, 0.0);
}

// Construtor com parâmetros (ii)
public Ponto2D(double x, double y) {
    this.x = x;
    this.y = y;
}
```



Características:

- Construtor default: inicializa o ponto na origem (0,0)
- Construtor parametrizado: permite criar pontos em qualquer posição
- Uso de `this()` para reutilização de código
- Uso de `this.x` e `this.y` para diferenciar atributos de parâmetros

Exemplos de uso:

```
Ponto2D origem = new Ponto2D();       // (0.0, 0.0)
Ponto2D p1 = new Ponto2D(3.0, 4.0);   // (3.0, 4.0)
```

Método calcularDistancia

Cálculo da distância euclidiana entre dois pontos

Implementação:

```
public double calcularDistancia(Ponto2D outroPonto) {
    double deltaX = this.x - outroPonto.getX();
    double deltaY = this.y - outroPonto.getY();
    return Math.sqrt(deltaX * deltaX + deltaY * deltaY);
}
```

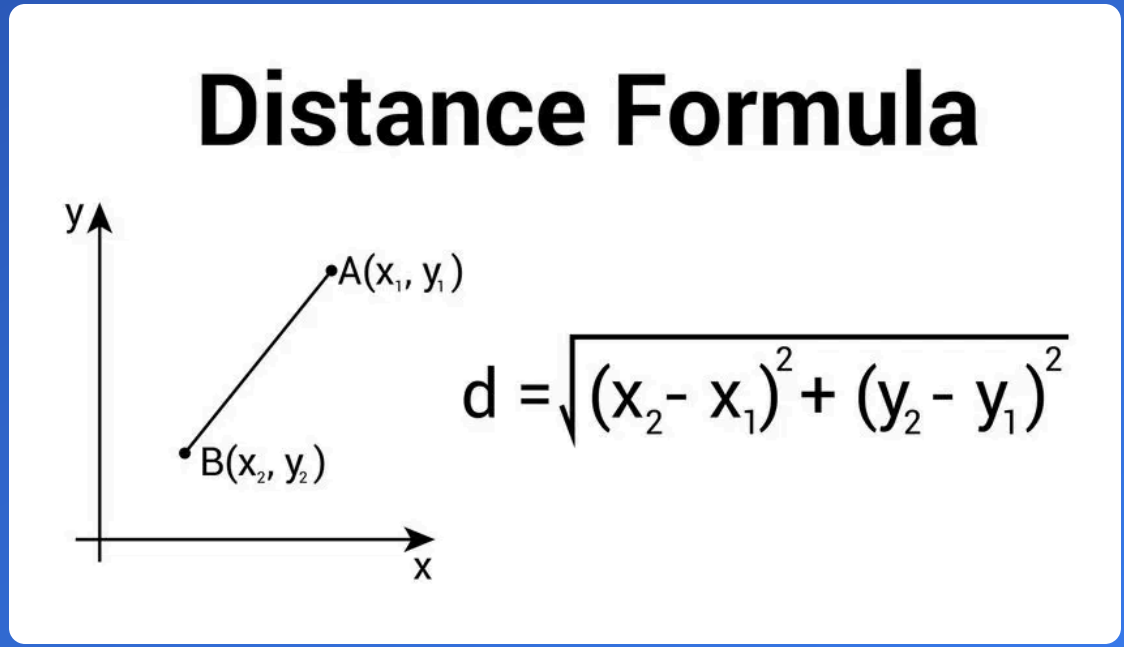
Fórmula da Distância Euclidiana:

A distância entre dois pontos (x_1, y_1) e (x_2, y_2) é calculada por:

$$d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$$

Onde:

- deltaX = $x_2 - x_1$
- deltaY = $y_2 - y_1$



Classe de Teste

Verificação das funcionalidades da classe Ponto2D

Casos de Teste:

```
public class TestePonto2D {
    public static void main(String[] args) {
        // Teste 1: Construtor default
        Ponto2D p1 = new Ponto2D();
        System.out.println("\n--- Teste 1 (Ponto default) ---");
        System.out.println("Ponto 1: " + p1);

        // Teste 2: Construtor com parâmetros
        Ponto2D p2 = new Ponto2D(3.0, 4.0);
        System.out.println("\n--- Teste 2 (Ponto com parâmetros) -
        System.out.println("Ponto 2: " + p2);

        // Teste 3: Calcular distância entre p1 e p2
        System.out.println("\n--- Teste 3 (Distância entre P1 e P2
        System.out.println("Distância entre P1 e P2: " +
                           p1.calcularDistancia(p2));

        // Teste 4: Calcular distância entre dois pontos arbitrári
```

Resultados dos Testes:

```
--- Teste 1 (Ponto default) ---
Ponto 1: (0.0, 0.0)

--- Teste 2 (Ponto com parâmetros) ---
Ponto 2: (3.0, 4.0)

--- Teste 3 (Distância entre P1 e P2) ---
Distância entre P1 e P2: 5.0

--- Teste 4 (Distância entre P3 e P4) ---
Ponto 3: (1.0, 1.0)
Ponto 4: (4.0, 5.0)
Distância entre P3 e P4: 5.0

--- Teste 5 (Distância de P2 para P2) ---
Distância de P2 para P2: 0.0
```

Análise dos Resultados:

- Construtor default inicializa corretamente na origem (0,0)
- Construtor parametrizado atribui valores corretamente
- Método toString formata corretamente as coordenadas
- Cálculo da distância euclidiana está correto (5.0 para os casos testados)
- Distância de um ponto para ele mesmo é zero, como esperado

Resultados e Conclusões

Análise dos testes e funcionamento da classe Ponto2D

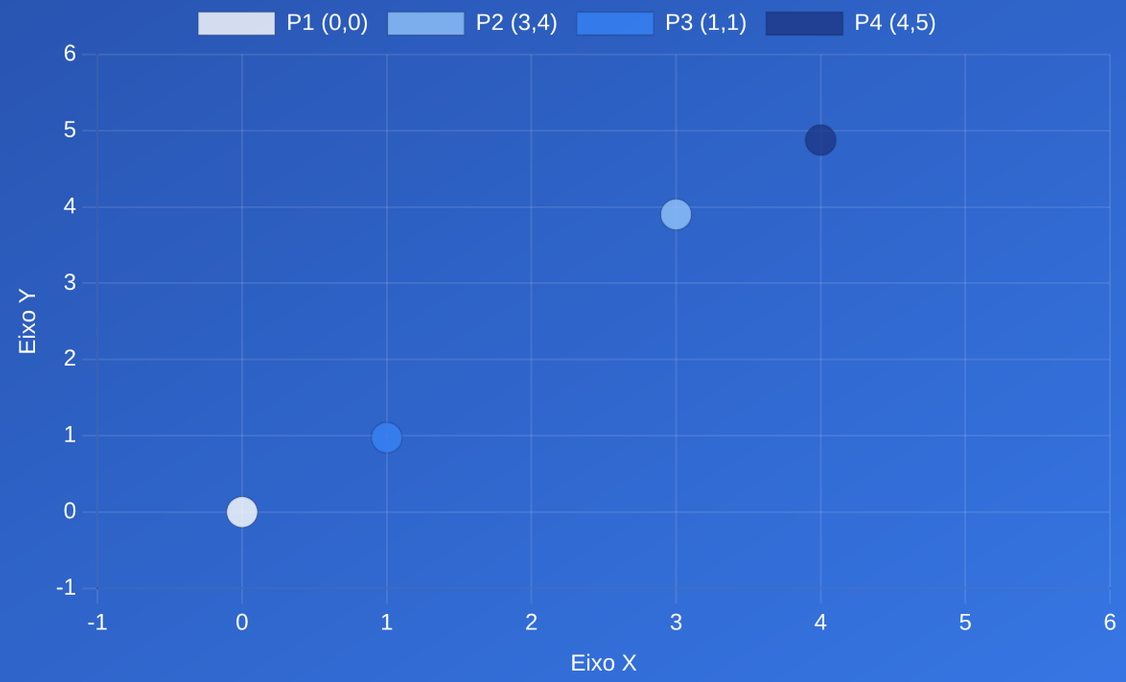
Resultados dos Testes:

Teste	Pontos	Distância
Ponto Default	P1 = (0.0, 0.0)	-
Ponto com Parâmetros	P2 = (3.0, 4.0)	-
Distância P1 a P2	P1 a P2	5.0
Distância P3 a P4	P3(1.0, 1.0) a P4(4.0, 5.0)	5.0
Distância ao mesmo ponto	P2 a P2	0.0

Conclusões:

- A classe implementa corretamente as coordenadas de um ponto 2D
- Os construtores funcionam conforme esperado
- O cálculo da distância euclidiana está correto
- A representação textual dos pontos é clara e intuitiva

Visualização dos Pontos no Plano Cartesiano



Aplicações:

A classe Ponto2D pode ser utilizada como base para diversas aplicações:

- Sistemas de coordenadas geográficas
- Cálculos geométricos em aplicações gráficas
- Base para classes mais complexas como polígonos
- Algoritmos de proximidade e clustering