

PG II

Programação Orientada por Objectos em Java

Associação de Classes:

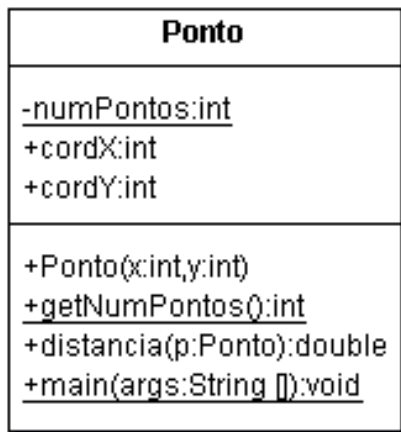
- Classe Ponto
- Classe Triangulo
- Métodos e Variáveis de Classe – Estáticos
- Classe IO
- Poligono

Classe Ponto

Aplicando os princípios de **OOP**, introduzidos até ao momento, será definida uma **classe Ponto** para representação de pontos num sistema de 2 eixos ortogonais, com a seguinte especificação:

- **Estrutura:** **cordX** e **cordY**, do tipo inteiro;
- Variável de classe (estática), **numPontos**, para contabilizar o número de pontos instanciados. Esta variável deverá ser *private*, existindo um método específico para consulta do seu valor.
- **Comportamento:** método de instância **distancia(Ponto p)**, para calcular a distancia do ponto "p" recebido como parâmetro ao "próprio" objecto.

Tal como esta forma de representação, outras poderiam ser usadas para especificar a implementação desta classe. Existe no entanto, uma convenção universal designada, **UML - Unified Modeling Language** – que reúne um conjunto de regras *standard* para a especificação, visualização, construção e documentação de arquitecturas de software. Segundo esta norma a classe Ponto é representada em UML, da seguinte forma:



- **Sinal "+" ou "-"** – Indica se aquele membro deve ser declarado como *public* ou *private*, respectivamente;
- **Sublinhado** – Indica que deve ser declarado como *static*;
- **...:<tipo>** - Tipo de uma variável ou tipo de retorno de um método/função.

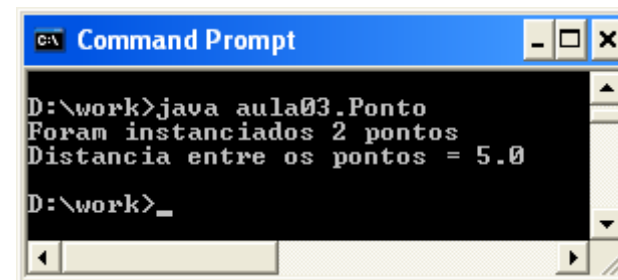
...Classe Ponto

Com base na especificação UML, a classe Ponto pode ser implementada em Java do seguinte modo:

```

1 package aula03;
2 import pg2.io.IO;
3 public class Ponto {
4     private static int numPontos;
5     public int cordX, cordY;
6     public Ponto(int x, int y) {
7         cordX = x;
8         cordY = y;
9         numPontos++;
10    }
11    public static int getNumPontos(){return numPontos;}
12    public double distancia(Ponto p){
13        double dx = cordX - p.cordX;
14        double dy = cordY - p.cordY;
15        return Math.sqrt( dx*dx + dy*dy );
16    }
17    public static void main(String [] args){
18        Ponto p1 = new Ponto (2,3);
19        Ponto p2 = new Ponto (5,7);
20        IO.cout.writeln("Foram instanciados " + Ponto.getNumPontos() + " pontos");
21        IO.cout.writeln("Distancia entre os pontos = " + p1.distancia(p2));
22    }
23 }

```



```

C:\ Command Prompt
D:\work>java aula03.Ponto
Foram instanciados 2 pontos
Distancia entre os pontos = 5.0
D:\work>

```

Classe Triangulo

Um Triangulo, que é formado por 3 pontos num sistema de 2 eixos ortogonais, pode ser representado por uma classe com a estrutura apresentada no seguinte modelo:

Triangulo
+cordX1:int
+cordY1:int
+cordX2:int
+cordY2:int
+cordX3:int
+cordY3:int

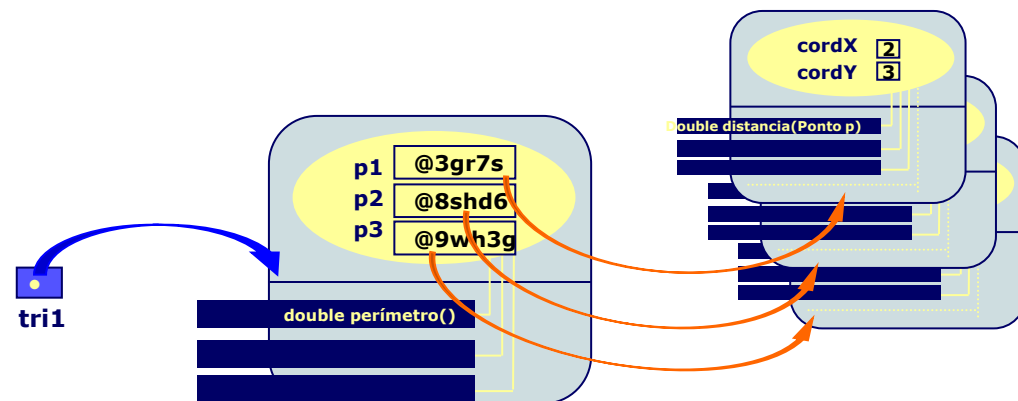
Cada um dos pares de valores cordX..., cordY..., representam as coordenadas de cada um dos pontos que formam o Triangulo.

Mas se:

- já existe uma classe que tem como objectivo representar um Ponto e as suas coordenadas;
- um dos princípios dados é a reutilização,

será que não existe outra forma mais expedita de implementar a classe Triangulo?

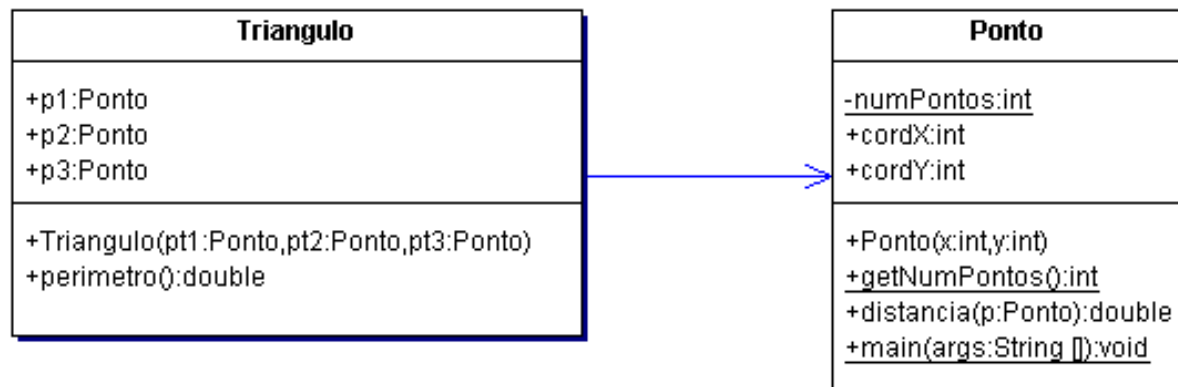
Outra forma de implementar a classe Triangulo é definir uma estrutura baseada em 3 variáveis de instância do tipo Ponto, onde cada uma irá guardar uma referência para uma determinada instancia da classe Ponto.



...Classe Triangulo

Além da estrutura proposta, será ainda adicionado ao comportamento da classe Triangulo, um método `perimetro()`. A especificação final desta classe fica então da seguinte forma:

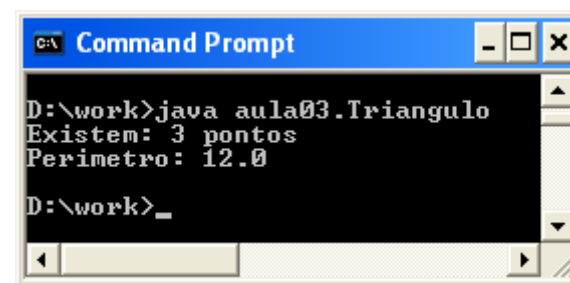
Neste diagrama a seta, indica a existência de uma associação entre a classe Triangulo e a classe Ponto. O sentido da seta, é o sentido da associação, ou seja, da classe que "conhece" para a classe "conhecida".



De notar que a implementação do método `perimetro()` é feita à custa do método `distancia()` dos objectos da classe Ponto.

```

1 package aula03;
2 import pg2.io.IO;
3 public class Triangulo {
4     public Ponto p1, p2, p3;
5     public Triangulo (Ponto pt1, Ponto pt2, Ponto pt3){
6         p1=pt1; p2=pt2; p3=pt3;
7     }
8     public double perimetro(){
9         double d = p1.distancia(p2);
10        d += p2.distancia(p3);
11        d += p3.distancia(p1);
12        return d;
13    }
14    public static void main(String [] args){
15        Ponto p1 = new Ponto (2,3), p2 = new Ponto (5,7), p3 = new Ponto(5,3);
16        Triangulo tr1 = new Triangulo (p1, p2, p3);
17        IO.cout.writeln ("Existem: " + Ponto.getNumPontos() + " pontos" );
18        IO.cout.writeln ("Perimetro: " + tr1.perimetro());
19    }
20 }
    
```



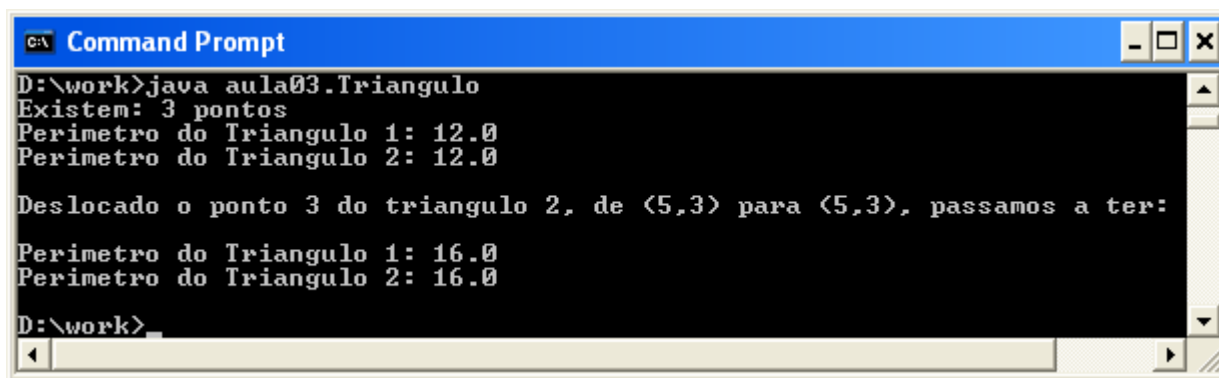
```

C:\ Command Prompt
D:\work>java aula03.Triangulo
Existem: 3 pontos
Perimetro: 12.0
D:\work>
    
```

Manipulação de referências para Objectos

Analise-se agora o seguinte algoritmo de teste da Classe Triangulo:

```
public class Triangulo {
    ...
    public static void main(String [] args){
        Ponto p1 = new Ponto (2,3), p2 = new Ponto (5,7), p3 = new Ponto(5,3);
        Triangulo tri1 = new Triangulo (p1, p2, p3);
        Triangulo tri2 = new Triangulo (p1, p2, p3);
        IO.cout.writeln("Existem: " + Ponto.getNumPontos() + " pontos" );
        IO.cout.writeln("Perimetro do Triangulo 1: " + tri1.perimetro());
        IO.cout.writeln("Perimetro do Triangulo 2: " + tri1.perimetro());
        tri2.p3.cordX=8;
        IO.cout.writeln("\nDeslocado o ponto 3 do triangulo 2, de (5,3) para (5,3), passamos a ter:\n");
        IO.cout.writeln("Perimetro do Triangulo 1: " + tri1.perimetro());
        IO.cout.writeln("Perimetro do Triangulo 2: " + tri1.perimetro());
    }
}
```



```
C:\> Command Prompt
D:\work>java aula03.Triangulo
Existem: 3 pontos
Perimetro do Triangulo 1: 12.0
Perimetro do Triangulo 2: 12.0

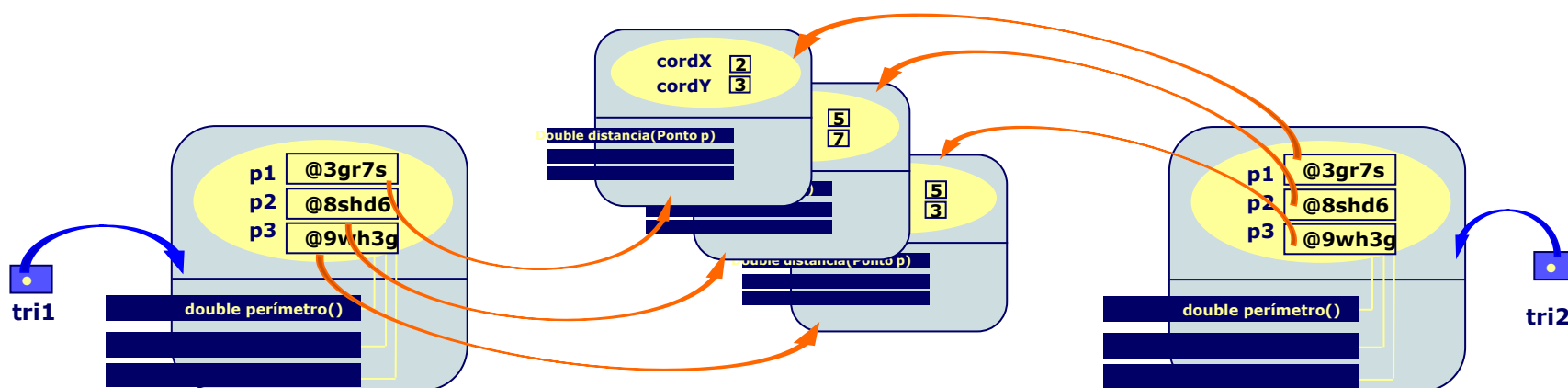
Deslocado o ponto 3 do triangulo 2, de <5,3> para <5,3>, passamos a ter:

Perimetro do Triangulo 1: 16.0
Perimetro do Triangulo 2: 16.0
D:\work>
```

Não tendo sido modificado o triangulo 1, porque é que o seu perímetro aparece alterado?

... Manipulação de referências para Objectos

Na realidade os dois triângulos foram criados com base no mesmo conjunto de pontos, sendo a situação final igual à que a seguir se apresenta:



Logo, deslocar o ponto 3 do triangulo 2 é o mesmo que deslocar o ponto 3 do triangulo 1. Ou seja, as expressões seguintes são equivalentes:

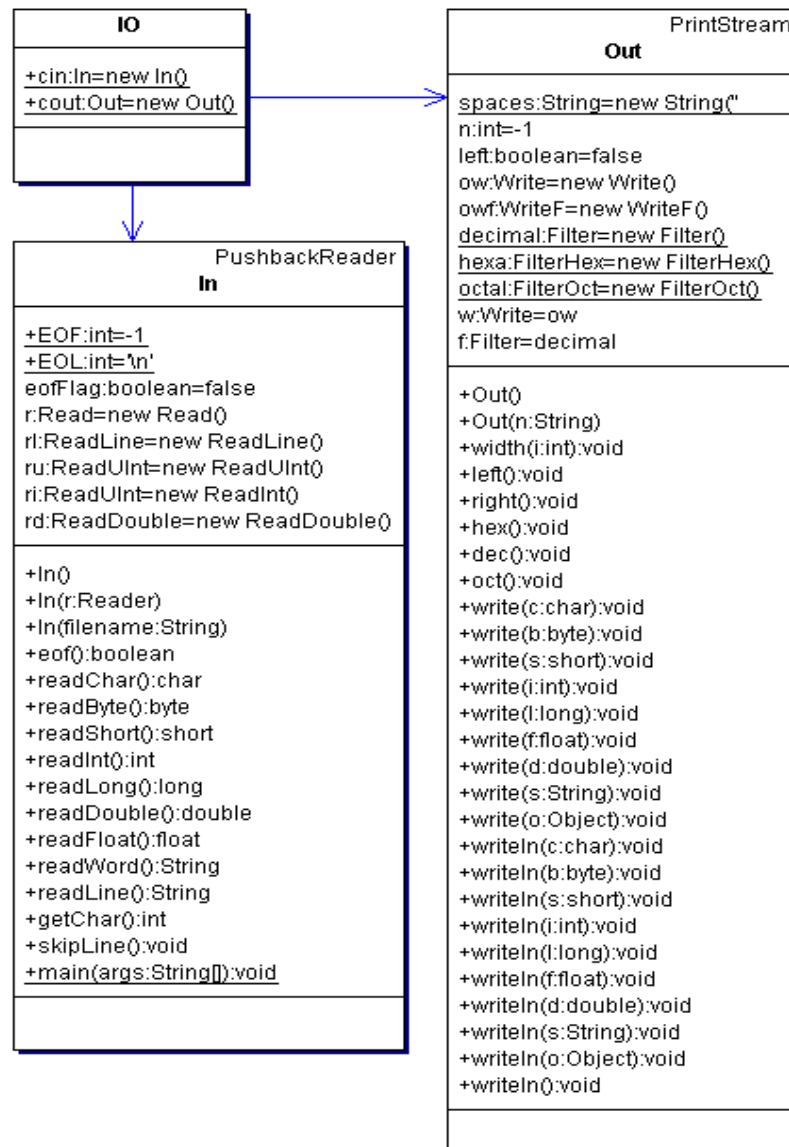
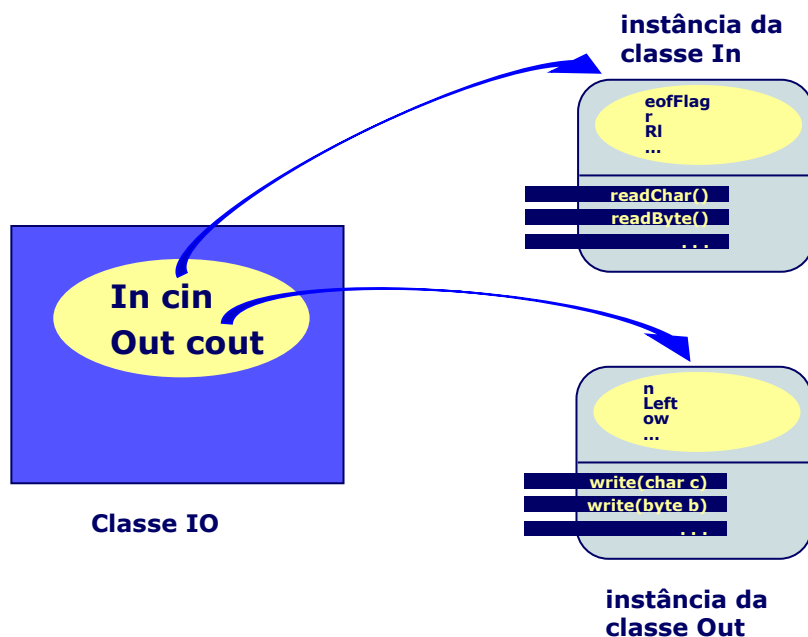
```
tri2.p3.cordX=8; ⇔ tri1.p3.cordX=8;
```

Métodos e Variáveis de Classe - Estáticos

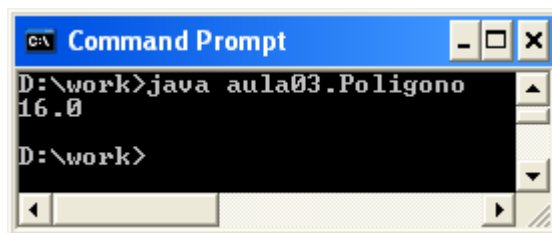
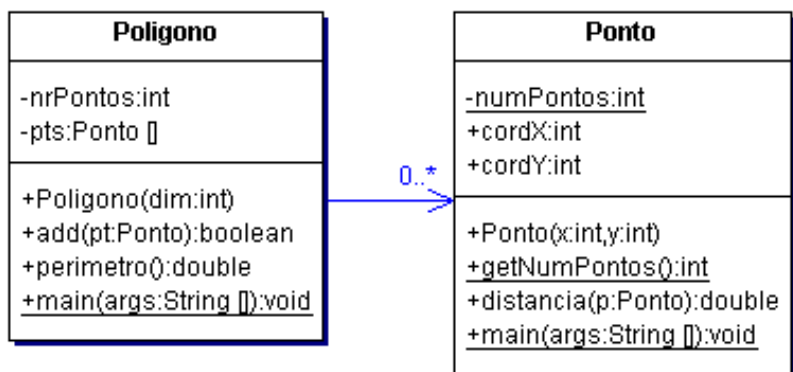
Conhecido o conceito de "static" e de "associação", já é possível entender a definição da classe IO:

```
package pg2.io;

public class IO {
    public static In cin = new In();
    public static Out cout = new Out();
}
```



Uma outra forma de simular a estrutura e comportamento de um triângulo é através de uma classe mais genérica que represente polígonos, como a que se apresenta na especificação seguinte:



```

1 package aula03;
2 import pg2.io.IO;
3 public class Poligono{
4     /*»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»*/
5     /*»»»»»»»»»»»»»»»»»»»» Variáveis de Instancia ««««««««««««««««««««««««/*
6     /*««««««««««««««««««««««««««««««««««««««««««««««««««««««««/*
7     private int nrPontos;
8     private Ponto [] pts;
9     /*»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»*/
10    /*»»»»»»»»»»»»»»»»»»»» Construtor      ««««««««««««««««««««««««/*
11    /*«««««««««««««««««««««««««««««««««««««««««««««««««««««««/*
12    public Poligono(int dim){ pts = new Ponto[dim]; nrPontos = 0;}
13    /*»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»*/
14    /*»»»»»»»»»»»»»»»»»»»» Métodos de Instancia ««««««««««««««««««««««/*
15    /*«««««««««««««««««««««««««««««««««««««««««««««««««««««««/*
16    public boolean add(Ponto pt){
17        if(nrPontos == pts.length) return false;|
18        pts[nrPontos++] = pt;
19        return true;
20    }
21    public double perimetro(){
22        double total=0;
23        for(int i=0; i<pts.length-1; i++)
24            total += pts[i].distancia(pts[i+1]);
25        total += pts[pts.length-1].distancia(pts[0]);
26        return total;
27    }
28    /*»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»*/
29    /*»»»»»»»»»»»»»»»»»»»» MAIN              «««««««««««««««««««««««/*
30    /*«««««««««««««««««««««««««««««««««««««««««««««««««««««««/*
31    public static void main(String [] args){
32        Poligono pol = new Poligono(3);
33        pol.add(new Ponto(2,3)); pol.add(new Ponto(5,7)); pol.add(new Ponto(8,3));
34        IO.cout.writeln(pol.perimetro());
35    }
36 }

```